AFRL-IF-RS-TR-2006-355
Final Technical Report
December 2006

OPERATIONAL INFORMATION MANAGEMENT SECURITY ARCHITECTURE

ITT Industries

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Air Force Research Laboratory Rome Research Site Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-IF-RS-TR-2006-355 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
DALE W. RICHARDS Work Unit Manager	JAMES W. CUSACK Chief, Information Systems Division Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
DEC 2006	Final	Mar 05 – Sep 06

4. TITLE AND SUBTITLE

OPERATIONAL INFORMATION MANAGEMENT SECURITY ARCHITECTURE

5a. CONTRACT NUMBER
FA8750-05-C-0105

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Vic Choo, Carol Muehrcke and Rob Vienneau

5d. PROJECT NUMBER
JBIS

5e. TASK NUMBER
00

5f. WORK UNIT NUMBER
01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
ITT Industries
Advanced Engineering and Sciences
775 Daedalion Drive
Rome NY 13441

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFSE
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2006-355

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 06-802

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This effort developed and demonstrated a basic security architecture for the Operational Information Management (OIM) project (previously known as Joint Battlespace Infosphere (JBI), with a particular focus on authentication and authorization. New security techniques, concepts of user privileges and access policies were investigated to support efficient and accreditable access control in a multi-level, secure environment implemented using a OIM-based infrastructure. Emphasis was on future compatibility with Net-Centric Enterprise Services (NCES) and Global Information Grid Enterprise Systems (GIG-ES) protocols, policies and processes for secure sharing of information between tactical assets, Command and Control (C2) platforms and intelligence, Surveillance and Reconnaissance (ISR) systems connected via an OIM infrastructure, as well as compliance with Director of Central Intelligence Directive (DCID) 6/3 guidance and requirements. The architecture specification includes a series of flow diagrams to show how information enters and propagates through the security components. The intent of the architecture design is not to prescribe how to implement each module, rather it shows what steps are necessary for the architecture to function properly. As part of the architecture development process, a methodology for assessing the risk associated with the architecture was also defined. The resulting architecture recommendations were demonstrated for a small OIM Reference Implementation instance and covered authentication and authorization, security policy management, and access control for increasing levels of security.

15. SUBJECT TERMS

JBI, OIM, Security, Architecture, NCES, GIG

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dale W. Richards
U	U	U	UL	137	19b. TELEPHONE NUMBER (*Include area code*)

Table of Contents

List of Figures

List of Tables

1.0 Executive Summary

This effort was initiated under management of the AFRL Joint Battlespace Infosphere (JBI) program. During the course of this effort that program evolved into the AFRL Operational Information Management (OIM) program. Thus, this effort which began as the JBI Security Architecture program itself evolved into the OIM Security Architecture to more properly reflect the updated vision and emphasis of AFRL information management research. The core JBI capabilities previously and collectively referred to as the Information Management Core System (IMCS) remain a critical and assumed component of the security architecture.

The primary objective of this effort was to define and demonstrate a security architecture applicable to the OIM, with a particular focus on authentication and authorization. The architecture development followed a requirements-driven process - where the requirements were defined and then an architecture was developed to meet those requirements. Of particular importance was maintaining compatibility with Global Information Grid (GIG) Information Assurance (IA) goals. The resulting architecture also makes use of security design patterns where applicable. These design patterns are considered "industry standards" for the handling of specific issues. The standards are based upon proven techniques, so they provide an element of risk reduction. The architecture specification includes a series of flow diagrams to show how information enters and propagates through the security components. The intent of the architecture design is not to prescribe how to implement each module, rather it shows what steps are necessary for the architecture to function properly. As part of the architecture development process, a methodology for assessing the risk associated with the architecture was also defined. The secondary objective of this effort was to develop a software prototype to demonstration and illustrate some of the key concepts of the proposed security architecture.

The OIM security architecture consists of modules that address authentication, authorization, administration, computer network defense, and federation. The output of this effort forms one portion of the security design required to protect the OIM as it matures. Additional security components addressing confidentiality, integrity, and availability are also required. The OIM security architecture is built around the Global Information Grid (GIG) concepts of policy decision and policy enforcement points. For both authentication and authorization, the architecture allows for the use of internal and external data stores.

2.0 Introduction

The Operations Information Management (OIM) program has demonstrated the value of a publish/subscribe/query capability as a network centric warfare (NCW) enabling technology. As the dependence on NCW increases, the need for an effective security architecture also becomes more important. Planning the security architecture that can be built into the system provides significant advantages over adding in the capability later. The primary focus of this report is a security architecture for the OIM that focuses on the authentication and authorization aspects. Other important subject areas, such as confidentiality, integrity, and availability are outside the scope of the effort and are only addressed at a cursory level.

The architecture was developed based upon a series of generated requirements and was developed without regard to specific products. The architecture also demonstrates compliance with the Global Information Grid (GIG) and Network Centric Enterprise Services (NCES) for future compatibilities.

2.1 Background

The OIM has several security features built in, including[1]:

- Hierarchical group constructs
- Security policy enumeration using the eXtensible Access Control Markup Language (XACML), access mapping and delegate interface
- Content-based access control
- Pluggable security architecture
- Secure socket layer connections

The security that is built into the system represents a good start. It covers the fundamental issues that need to be addressed. Applying the approaches described in this report will provide enhancements to address the future needs of supporting net-centric operations.

This document describes the key drivers for the architecture design. This includes a summary of functionality and programmatic considerations. From there, we describe the key components of the architecture starting at a high level and then at progressively lower levels. We describe interfaces to ensure compatibility with other GIG components. We then describe the rationale behind the architecture. Finally, we address other security issues at a cursory level.

[1] Clark, Thomas, "Core Services Reference Implementation", JBI Program Review, March 2005.

2.2 Scope

This project has evolved since its conception. Initially, the effort was aimed at providing relatively high level architecture for a number of key information assurance areas including authentication, authorization, confidentiality, integrity and availability. The focus then shifted to a more detailed look at the authentication and authorization components. The document also extends the security coverage to the federated case. The document describes the architecture in terms of OIM clients and servers, but does not address its use relative to OIM fuselets.

2.3 Methodology

Our methodology focused on learning from the past, and looking towards the future. As such, we performed the following:

- Identified high level drivers and decisions related to the architecture (Section **Error! Reference source not found.**)
- Studied existing architectures (GIG, NCES, security design patterns, Knowledgeable Agents-oriented System (KAoS) from the Institute for Human and Machine Cognition, commercial federation architectures and Secure Socket Layer (SSL)) (Section 7.1)
- Generated rough requirements
- Identified process flows
- Generated flow diagrams (Section **Error! Reference source not found.**)
- Documented architecture and rationale (Sections 3 - **Error! Reference source not found.**)
- Developed a demonstration capability

2.4 Additional Documentation

There are several additional unpublished documents provided to AFRL under this effort:

1. "OIM Security Architecture – Project Documentation," 30 June 2006. This report has complete appendices that cover the following: Acronym List; Definition of Terms; DCID 6-3 Requirements for Gemini; Information Core Services Security Architecture; KAoS Startup Notes; Network Security Standards Organizations; Security Markup Languages; Standard Security Frameworks; JBI Coalesced Requirements; and Mission Oriented Risk and Design Analysis (MORDA).
2. "OIM Software Prototype Test Procedures," 30 August 2006.
3. "OIM Security Architecture Software Document," 30 August 2006.
4. "OIM Security Architecture Demonstration User's Guide and Demonstration Script," 30 August 2006.

3.0 Requirements

A requirements analysis was performed to establish a series of goals for the security architecture. At the time this effort was underway, portions of the Global Information Grid Information Assurance requirements were still under development. There did not appear to be any formalized net-centric security requirements. It was believed that the Air Force Warfighting Integration office, AF/XI, is in the process of developing such a specification, but there was no verification that the document exists.

3.1 DCID 6/3 Accreditation

The Directorate of Central Intelligence Directive (DCID) 6/3 Protecting Sensitive Compartmented Information within Information Systems analysis focused on identifying topic areas that should be considered in order to help simplify a potential future accreditation request. This particular task extended the DCID 6/3 analysis another step and established the security requirements for the OIM Security Architecture. This assessment developed a series of coalesced requirements from a number of different sources from within the DoD and industry. It should be noted that the requirements are considered to be a "living document". These requirements do not address every known security issue. That is, they are anticipated to morph as new threats, technologies, services, and approaches evolve. A summary of this analysis is presented next. The complete listing of the coalesced security requirements and where they are addressed in the OIM SA is provided in the project documentation report [2].

3.1.1 DCID 6/3 Accreditation Requirements

The objective of this study was to determine what steps to prudently take now to minimize the impacts of the accreditation process on future OIM versions. The initial study focused on generating a requirements specification for the OIM Gemini program. A summary of this study is provided below.

The Director of Central Intelligence Directive 6/3 (DCID 6/3), *Protecting Sensitive Compartmented Information Within Information Systems*[3], defines certification and accreditation requirements for information systems that process, store, or communicate classified intelligence information. Our team analyzed the DCID 6/3 specification to determine requirements for accrediting OIM under this document. Table 1 counts DCID 6/3 requirements in areas identified to be of concern for OIM.

---

[2] "OIM Security Architecture Project Documentation," 30 June 2006.

[3] Director of Central Intelligence Directive 6/3, *Protecting Sensitive Compartmented Information Within Information Systems*. 24 May 2000

Table 1 - OIM Security Requirements Summary by Area of Concern:[4]

Area of Concern	Number
Named Attacks and Vulnerabilities	10
General	24
Identification and Authentication	16
Access Control	14
Account Management	6
Multiple Security Levels	20
Backups and Recovery	13
Auditing	20
Malicious Code Detection	18
Configuration Management and Change Control	11
Documentation	16
Total	168

The initial process for analyzing DCID 6/3 requirements was suggested by the *System Security Authorization Agreement (SSAA) for the Certification and Accreditation of Infrastructure Operations Tools Access* (IOTA). IOTA's DCID 6/3 accreditation was based on testing performed by the DoD Intelligence Information Systems (DODIIS) Independent Test Authority (ITA). The SSAA for IOTA lists DCID 6/3 requirements that it meets, while classifying others as a site responsibility or not applicable. This suggested classifying DCID 6/3 requirements into three categories: requirements that OIM can influence; requirements that OIM cannot influence; and requirements that are not applicable. Table 2 lists the number of requirements that fall into each of these categories.

[4] "*Suggested Security Certification and Accreditation Requirements for Gemini Specification*", ITT Industries, 28 July 2005

Table 2 - OIM Security Requirements Summary by DCID 6/3

Chapter	Level/Area	Influenced By OIM	OIM Cannot Influence	Not Applicable
Confidentiality	PL1	27	12	6
	PL2	11	10	1
	PL3	8	2	1
	PL4	25	4	0
	PL5	5	1	0
Integrity	Basic	5	1	0
	Medium	6	6	0
	High	9	6	0
Availability	Basic	2	1	0
	Medium	9	6	0
	High	7	6	0
Advanced Technology	Interconnected IIS	1	4	0
	Controlled Interface	1	42	1
	Web Security	5	1	0
	Servers	3	9	0
	Mobile Code	6	2	0
	E-Mail	3	2	0
	Collaborative Computing	0	11	0
	Distributed Processing	0	1	0
Administration	Procedural	0	153	0

	Environmental	1	7	0
	Physical	0	3	0
	Personnel	0	1	0
	Foreign Nationals	0	3	0
	Handling Restrictions	0	1	0
Risks, Etc.	Overview	0	4	0
	Risk Management	0	22	0
	Certification	0	5	0
	Accreditation	0	55	0
	C&A Process	5	11	0
	Exceptions	0	13	0
	Special IS Categories	0	18	10
Total		139	423	19

This grouping of requirements is not convenient for OIM development concerns. We reorganized DCID 6/3 requirements by OIM functional areas (Table 1) and commented on the implications for OIM of applicable requirements. A gap analysis of these DCID 6/3 requirements with OIM Mercury implementations suggested the following features offer a significant increment toward an accreditable system:

- Authentication and encryption between configurable clients and OIM core servers (as in Mercury), among OIM core servers, and among modules implementing a distributed OIM core server. Secure Sockets Layer (SSL) provides an implementation for these capabilities. All servers would have a certificate, and this is needed eventually to work in the NCES environment.
- Initial capabilities for fuselet security, privileges to run fuselets, and the checking of fuselet digital signatures.
- Auditing and monitoring capabilities allowing real-time analysis. Gemini should not only provide an audit architecture, but also start to analyze how attacks on the OIM would look and to define specific audit events to be collected to provide an ability to detect these attacks. Denial Of Service (DOS) attacks and suspicious fuselet activity are

key concerns. This could be a valuable addition to earlier OIM efforts on instrumentation and control. Availability is going to trump many other concerns about this system.

- The determination of whether persistent subscriptions (that live across logins) are an operational requirement, considering logoff and session termination requirements in DCID 6/3.
- Backup and restore capabilities, if this can be done in an efficient way independent of the underlying database technologies.

The detailed results of the DCID 6/3 analysis are provided in Appendix III of the OIM Security Architecture Project Documentation report.[5]

3.2 Coalesced Requirements

The coalesced requirements represent a consolidated listing of specifications, requirements, and standard practices that were gathered from the documents listed in Table 3 - Sources for Coalesced Requirements

Table 3 - Sources for Coalesced Requirements

Document Title	Date
NIST 800-53 Recommended Security Controls for Federal Information Systems	Feb-05
NIST 800-23 Guidelines to Federal Organizations on Security Assurance and Acquisition/Use of Tested/Evaluated Products	Aug-00
DCID 6/3 Protecting Sensitive Compartmented Information Within Information Systems - Manual	May-00
DODI 8500.1 Information Assurance	Nov-03
DODD 8500.2 Information Assurance	Nov-03
DOD Information Technology Security Certification and Accreditation Process (DITSCAP)	Jul-00
DODI 8320.2 Data Sharing in a Net-Centric Department of Defense	Dec-04
DOD Discovery Meta Data Specification, V 1.2	Jan-05
DODI 8551.1 Ports, Protocols, and Services Management	Aug-04
DOD 8520.1 DOD Protection of Sensitive Compartmented Information	Dec-01
DOD IA Strategic Plan, V 1.10	Jan-04
DODI 8110.1 Multinational Information Sharing Network Implementation	Feb-04
CJCSI-6212.1C Interoperability and Supportabilty of Information	Nov-03

[5] "OIM Security Architecture Project Documentation," 30 June 2006.

Technology and National Security Systems	
CJCSI 6510.01D Information Assurance and Computer Network Defense	Jun-04
Common Criteria User's Guide	Oct-99
Common Criteria Part 1: Introduction and General Model, V 2.2	Jan-04
Common Criteria Part 2: Security Functional Requirements	Jan-04
Common Criteria Part 3: Security Assurance Requirements	Jan-04

The coalesced requirements document lists relevant security topics, including:

- Identification
- Authentication
- Authorization (permissions, policies)
- Federation aspects
- Confidentiality (e.g., privacy)
- Integrity (e.g., data protection)
- Availability (e.g., quality of service)
- Software development

Together, the coalesced requirements for these topic areas served as a starting point for the design of the architecture.

In general, the identified requirements were divided into the following three areas: functional, design, and implementation. Each of these categories corresponds to a specific stage in the architecture development process. Within each category, the requirements were further divided into sub-categories as shown in Table 4.

Table 4 - Coalesced Requirements Areas

Functional	Design	Implementation
General	General	General
Threats	Identification & Authentication	Identification & Authentication
Identification & Authentication	Integrity/Confidentiality	• Passwords
• Policies	Availability	• PKI
Integrity/Confidentiality		• Restricted Servers
• MSL/MLS		• Miscellaneous
Availability		Integrity/Confidentiality
Accountability		Malicious Code

Auditing		Acquisition Standards
Non-Repudiation		Reviews
Malicious Code		Compliance Testing
Data Sharing		System Assurance
		Configuration Mgmt.
		Maintenance
		Documentation

The majority of the requirements fell in the implementation area. Many of these implementation requirements are addressed by COTS and open source implementations. Requirements range from general to specific. For example, a general requirement is that a system must have a method of authentication, while a specific requirement is that passwords must contain at least eight characters and be a mix of upper case, lower case, numbers, and special characters. Some requirements of particular interest follow:

- Support shall be provided for multiple access control models
- Compatibility with IPv6 shall be provided
- Support for wireless environments shall be provided
- Support for active networks shall be provided
- Requirements for propagation and synchronization of data shall be met

4.0 Background Studies

A number of background studies were performed to identify relevant security markup languages as well as security organizations that are likely to provide inputs that are relevant to the security architecture.

4.1 Emerging Security Markup Languages and Web Services Specifications Investigations

The activities of security standards groups were examined to determine the status of their activities. This was primarily geared to developments with respect to XML variations and web services specifications. A series of summary reports on these topics were prepared to determine their applicability to the security architecture. A summary of these investigations are provided next.

4.1.1 Security Standards Organizations

Several groups were identified that are proposing standards for web service and security related protocols and approaches. The following provides a summary of these organizations and their activities. Additional information on these groups is provided in [6].

4.1.2 Internet Engineering Task Force (IETF)

The IETF is a consortium that is comprised of representatives from multiple organizations. The IETF has a number of working groups active at any one time. The goal of the working groups is to establish standards to address specific problem areas. Typically, the groups will develop a proposal for standards, whether a protocol definition, data exchange method, or procedure, and publish the results as a "request for comment" (RFC). The inputs to the RFC are incorporated into the proposal and the document provided to other groups for ratification as a standard. This process has a varied lifecycle which results in groups constantly forming and disbanding. A list of the various IETF working groups that cover security related areas is provided in Table 5.

Table 5 - IETF Security Related Working Groups

Acronym	Group
BTNS	Better-Than-Nothing Security
ENROLL	Credential and Provisioning
IDWG	Intrusion Detection Exchange Format
INCH	Extended Incident Handling
ISMS	Integrated Security Model for SNMP

[6] "OIM Security Architecture Project Documentation," 30 June 2006.

KINK	Kerberized Internet Negotiation of Keys
KITTEN	Kitten (GSS-API Next Generation)
KRB-WG	Kerberos Working Group
LTANS	Long-Term Archive and Notary Services
MOBIKE	IKEv2 Mobility and Multihoming
MSEC	Multicast Security
OPENPGP	An Open Specification for Pretty Good Privacy
PKI4IPSEC	Profiling Use of PKI in IPSEC
PKIX	Public-Key Infrastructure (X.509)
SACRED	Securely Available Credentials
SASL	Simple Authentication and Security Layer
SECSH	Secure Shell
SMIME	S/MIME Mail Security
SYSLOG	Security Issues in Network Event Logging
TLS	Transport Layer Security

A few of the groups have generated RFCs that are relevant to the OIM SA. Examples of these are listed in Table 6.

Table 6 - Examples of IETF Security Group Activity

Organization	Group	Recent Activity
IETF	AAA (Authentication, Authorization, Accounting)	Submitted several RFC (request for comments) proposals for Diameter base protocol for AAA.
IETF	SACRED (Secure Available Credentials)	Submitted frameworks and protocols as RFC.
IETF	SASL (Simple Authentication and Security Layer)	Submitted SASL to the IESG for consideration as a proposed standard.

4.1.3 Liberty Alliance

The Liberty Alliance Project is an alliance of more than 150 companies, non-profit and government organizations from around the globe. Created in 2001, the consortium is committed to developing an open standard for federated network identity that supports all current and emerging network devices. It is envisioned that Federated Identity management will provide substantial benefits to the end user in terms of managing accounts.

4.1.4 World Wide Web Consortium (W3C)

The World Wide Web Consortium (W3C) is an international consortium that develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its

full potential. W3C is a forum for information, commerce, communication, and collective understanding.

4.2 Security Languages

Various extensions to XML have been standardized in the last few years, and other extensions are being considered by standards committees. Under this effort, our team evaluated the use of emerging security markup languages, such as the Security Assertion Markup Language (SAML) and the eXtensible Access Control Markup Language (XACML) for application to the OIM SA.

Our team began by identifying several of the more popular markup languages, shown in Table 7. As a first stage in evaluating these languages, we took a quick look at each language. The quick look is a high level assessment of the technology to determine its capabilities, possible functional roles, and pros and cons. The purpose of the quick look was to determine the potential application of a technology, standard, language, etc. in the OIM Security Architecture. Furthermore, the quick look analysis also sought to determine the technologies that are moving forward and those that are not being supported. The "OIM Security Architecture Project Documentation" report documents the results of the high level assessment of these languages, including a list of the positive and negative aspects.

Table 7 - Standard Markup Language Examples

Markup Language	Standardization Status
Web Ontology Language (OWL)	A W3C recommendation
Security Assertion Markup Language (SAML)	V. 2.0 approved as an OASIS standard
eXtensible Access Control Markup Language (XACML)	An OASIS standard
XML Key Management Specification (XKMS)	An open specification published by the W3C
eXtensible rights Markup Language (XrML)	OASIS standardization suspended

OWL is a useful ontology language that appears to have some backing throughout the community. The OIM team has developed an initial OWL-based ontology.

XACML is considered to be a useful language extension and has many positive features, such as role based access control (RBAC) support. However, some implementations, such as Knowledgeable Agents-oriented System (KAoS) developed by the Institute for Human & Machine Cognition (IHMC), convert XACML into another internal language to support expressions not covered by XACML.

SAML is primarily used for passing authentication data. It does have some drawbacks as it doesn't work well when the authentication mechanism needs to transmit large amounts of data.

5.0 Key Architecture Drivers and Decisions

This section briefly describes the key drivers and decisions that shaped the proposed authentication and authorization architecture for OIM. A review of how the architecture reflects these points can be found below in Section 7.3.

- **Meet and exceed GIG Increment 1 requirements:** The architecture maintains consistency where possible with anticipated GIG Identification and Authentication and Policy Based Access Control architectures for Increment 1 (2008), while supporting capabilities particularly important for OIM applications that do not appear generally in the GIG plans until beyond 2008. These advanced capabilities are:
    - **Advanced authorization policies** based on rich and extensible sets of policy parameters
    - Authentication and authorization for **automated processes as clients**
    - Infrastructure for use of **authentication assurance level**
- **Use NCES Security Services or internal OIM security functions:** The architecture allows OIM to take advantage of the evolution of NCES, without being dependent upon it.
- **Provide certificate and password authentication, while allowing a path for support of other methods:** The architecture supports any authentication algorithm that is realized as a protocol exchange between a client and the server that serves to authenticate the client to the server. The protocol exchange may also include server to client authentication.
- **Support locally maintained policies and attributes, externally maintained policies and attributes, or a combination of both:** In-process GIG architecture studies indicate that in the future the authorization decision for a resource may depend upon policies and attributes that are maintained both internal and external to the OIM environment. The architecture provides the infrastructure to allow for this, even though concepts for aligning and reconciling external and internal attributes and policies are not presently well understood.
- **Provide transparent and secure access to data residing on any OIM instance (a.k.a. federation):** The OIM vision is that data anywhere in the network of OIM's can be accessed by those authorized to do so, if they have access to any OIM instance. The drivers of transparency and security lead us in turn to make security checks transparent and place the burden of federation processing on the OIM server, minimizing the impact on clients. This approach also provides greater leverage for development and assurance efforts.
- **Integrate advanced support for Computer Network Defense:** The architecture supports two distinct streams of data useful for computer network defense that arise from the authentication and authorization processes. These are data for real or near-real time situation analysis, and traditional audit data.
- **Operate both in web service and Application Programming Interface (API) mode:** Using the components of this architecture, a common OIM data repository can

simultaneously serve clients interested in a web service interface, or directly serve clients that do not have the capability to interact with a web service. This includes those that must use passwords for authentication instead of Public Key Incription (PKI) certificates. By “directly serve,” we mean that no intervening web portal is required to mediate between the client and OIM.

6.0 Architecture Overview

Figure 1 shows the modules and module interactions in the proposed architecture. OIM security modules are shown within the dashed boundary. This section provides an overview of the architecture:

- Section 0 describes the modules and module interactions at a high level.
- Section 6.2 describes interactions with external entities.
- Section 6.3 summarizes key characteristics of the architecture.
- Section 6.4 describes options provided by the architecture to support the needs of a variety of OIM applications, and how they are realized within the architecture.

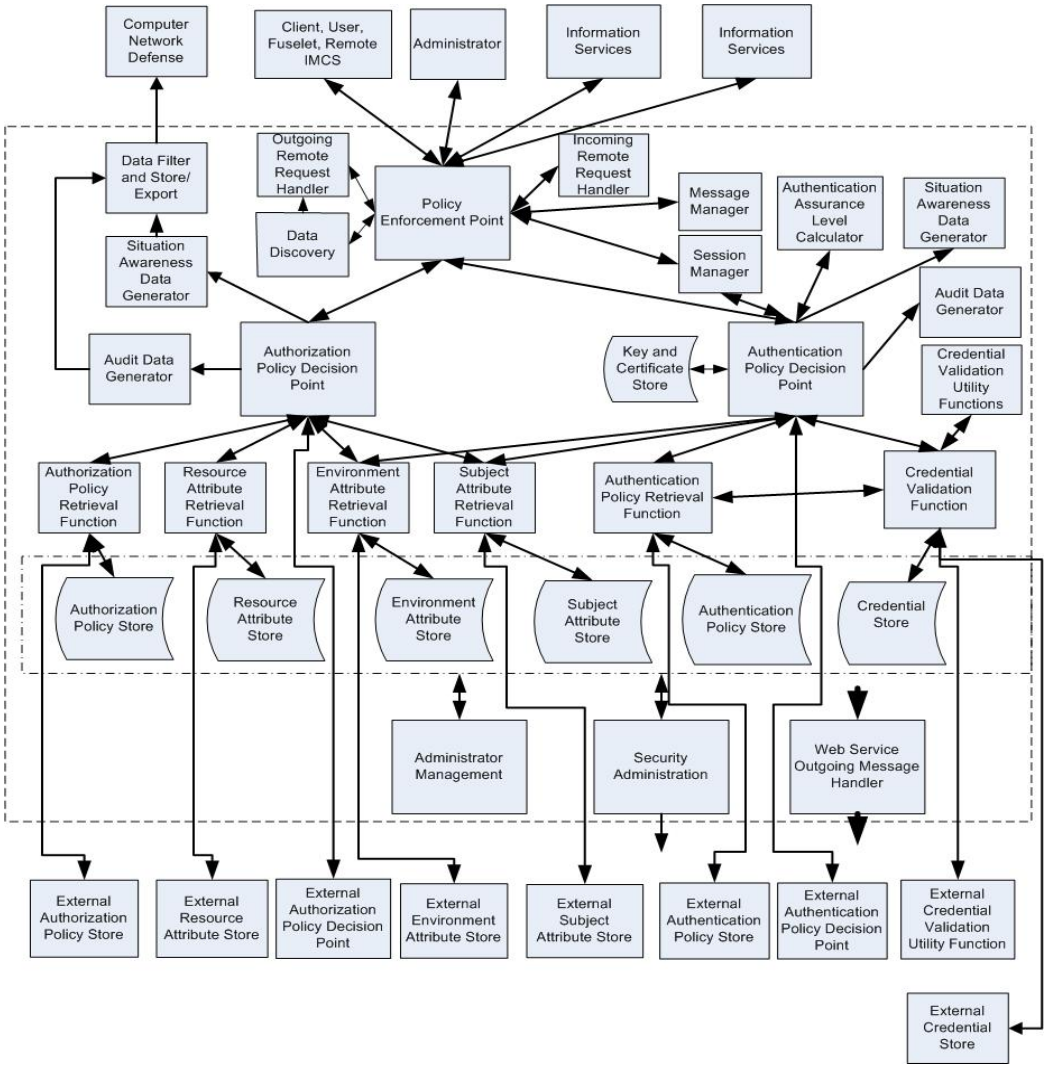

Figure 1 - OIM Authentication and Authorization Modules

Section 9.0 provides further details on the operation of the architecture by presenting detailed process flows for key processing sequences. Module List in Section 10.0 reviews the modules in list form, with brief descriptions.

6.1 OIM Modules

6.1.1 Architecture Backbone

The backbone of the architecture is the triangle consisting of the **Policy Enforcement Point**, the **Authentication Policy Decision Point** and the **Authorization Policy Decision Point**. The Policy Enforcement Point (PEP) is the single interface for handling incoming requests for OIM services. The PEP consults the Authentication Policy Decision Point to determine whether the system will accept the proof presented of the requester's identity. The PEP consults the Authorization Policy Decision Point to determine whether this requester is authorized for the action requested.

The OIM architecture supports both a login-based session model and a web services individual transaction-based model, so it supports authentication and authorization for both of these models. The **Session Manager** and the **Message Manager** support the Policy Enforcement Point by tracking individual active login sessions and web service request messages undergoing security processing.

6.1.2 Authentication Decisions

Both the Authentication Policy Decision Point and the Authorization Policy Decision Point use a set of retrieval functions and associated data stores to obtain the information upon which they base their decisions. The decision whether to accept a requester's proof of identity, in other words to authenticate a requester, is based on policy stored in an **Authentication Policy Store,** and retrieved by the **Authentication Policy Retrieval Function**. Such policy stores may exist either internally or externally to OIM, or both.

Application of an authentication policy consists of evaluating the proof of identity presented by the requester, and other attributes of the request scenario. The proof of identity presented by a requester may be considered adequate in some scenarios and not in others. The Authentication Policy Decision Point performs the authentication protocol exchanges required to implement supported authentication protocols. It relies on support from the **Credential Validation Function**, which coordinates processing required to examine specific types of credentials presented, such as matching a password or validating a certificate. The Credential Validation Function may rely on protocol specific internal or external services to assist in its function, called **Credential Validation Utility Functions** and **External Credential Validation Utility Functions**. As examples, a Credential Validation Utility Function or External Credential Validation Utility Function may perform certificate validation in the case of certificate based authentication, and digital signature checking if implementing NCES compliant authentication based on signed Simple Object Access Protocol (SOAP) messages. The NCES Certificate Validation Service is an example of an External Credential Validation Utility Function. Before

performing its check on a credential, the Credential Validation Function may need to retrieve the full credential if it received only a reference to it, from an internal or external **Credential Store**.

Once the requester's proof of identity has been examined, the next step before accepting an authentication is to consider attributes of the requester and the environment. The Authentication Policy Decision Point finds these attributes stored in the **Subject Attribute Store** and **Environment Attribute Store**. The **Subject Attribute Retrieval Function** and **Environment Attribute Retrieval Function** locate and retrieve the attributes required to evaluate the authentication policy.

The Authentication Policy Decision Point has access to all data required to calculate the authentication assurance level, which is a GIG-defined quantitative measurement of confidence in the requester's authentication calculated based on the authentication protocol and credentials, requester attributes and environment attributes. The Authentication Policy Decision Point makes these parameters available to the **Authentication Assurance Level Calculator**. The authentication assurance level value may be referenced in the authentication and/or the authorization policy.

6.1.3 Authorization Decisions

The decision whether to authorize a requester for the action requested is based on policy stored in an **Authorization Policy Store,** and retrieved by the **Authorization Policy Retrieval Function**. Such policy stores may exist either internally or externally to OIM, or both.

Authorization policies consider attributes of the resource to which access has been requested, in addition to subject and environment attributes. Some subject and environment attributes considered by the authorization policy may be the same as the attributes considered by the authentication policy. The Authorization Decision Point obtains resource attributes from the **Resource Attribute Store** via the **Resource Attribute Retrieval Function**. Some resource attributes such as file releasability are retrieved based on file metadata values, and may be maintained external to OIM.

6.1.4 Federation

Support for federated operations requires three unique modules, which are **Data Discovery**, the **Outgoing Remote Request Handler**, and the **Incoming Remote Request Handler**. Data Discovery determines if a requester requires services for data not managed by the OIM PEP which logged in the requester, and determines the location of this remote data. The Outgoing Remote Request Handler sends out a request to a remote OIM PEP, and the Incoming Remote Request Handler processes such requests from other OIM instances.

6.1.5 NCES Compatible Web Services Support

Support for sending outgoing messages to web services (such as NCES security services) requires the **Web Services Outgoing Message Handler**. This module packages outgoing

messages in the NCES-defined format, including application of the digital signature of this OIM instance. Note that functions within the generic authentication and authorization modules previously described handle incoming NCES compliant messages when OIM itself is operating as a web service, so a separate module is not proposed for this purpose.

6.1.6 Keys and Certificates

The **Key and Certificate Store** contains the private key and certificate for this OIM instance. These credentials are needed if an OIM instance performs any of the following functions:

- Sends a remote requests to a federated OIM instance, for authenticating itself to that federated OIM
- Sends a request to an external web service such as NCES security services, for authenticating itself to that web service
- Uses an authentication protocol that includes certificate based authentication of OIM to a client, such as SSL.

The Key and Certificate Store also may contain a pre-distributed NCES Certificate Validation Service certificate that is implicitly trusted. This is required for using the NCES Certificate Validation Service.

6.1.7 Administration

Although this cannot be shown conveniently in the figure, the **Security Administration** module has an interface to the Policy Enforcement Point, through which it accesses all internal policy stores and attribute stores. It provides the management interface to the data in these stores. The **Administrator Management** module provides a management interface to set up the privileges for the various administrators of the system. Similarly, it has an interface to the Policy Enforcement Point via which it accesses the Subject Attribute Store and Authorization Policy Store.

6.1.8 Data Generation for Computer Network Defense

The **Audit Data Generator** captures audit records of authentication and authorization decisions via its interface with the Authentication Decision Point and the Authorization Decision Point, respectively. The **Situation Awareness Data Generator** provides data about authentication and authorization events for any real time or near real time analysis process that monitors the security health of OIM, or of the GIG or portions of the GIG. The Audit Data Generator and Situation Awareness Generator feed information to the **Data Filter and Store/Export** Function, which determines how to filter, format and store and/or forward this data to internal stores and external recipients.

6.2 External Interfaces

Interfaces to OIM fall into several categories, enumerated below, and illustrated in Figure 2:

- **Client:** External entities that request OIM services. These may be a user, an automated program, or a federated OIM.
- **Administrator:** Interface for control of OIM configuration and security parameters.
- **Identification/authentication infrastructure:** Externally provided functions related to identifying and authenticating the entities that request OIM services, including certificate processing. It is anticipated that GIG/NCES will provide these functions.
    - **External Credential Store**– Contains credentials for lookup by OIM, where lookup parameters are preferred to be passed in by the client instead of a full credential. The known example is the NCES Certificate Retrieval Service, where a certificate can be retrieved based on an issuer and a serial number.
    - **External Credential Validation Utility Function**– function that assists in checking the validity of a credential. The key example is an NCES certificate revocation checking service.
- **Attribute stores:** External stores that contain attributes used by OIM in making policy decisions. There may be one or several stores of each type consulted by OIM. These stores are:
    - **External Subject Attribute Store** – contains attributes of entities that request OIM services. The key example is the GIG Identity Manager, which contains identity assurance level and active/suspended status for GIG users. Similar stores might exist to handle automated programs rather than human users.
    - **External Environment Attribute Store** - contains information about the environment, typically for subsets of the GIG, such as networks or individual servers or end user systems. An example might be "base INFOCON status", ALPHA, BRAVO, CHARLIE, or DELTA.
    - **External Resource Attribute Store** – contains attributes of resources that the OIM offers to clients that are not maintained within the OIM itself. Since the major type of resource attribute is metadata, this is maintained within OIM. Looking forward, there could be attributes maintained outside of OIM, for example quality of protection or releasability values that are determined based on OIM-maintained metadata values.
- **Policy-Related:** External stores that contain policy data used by OIM when making policy decisions, and external services that directly provide policy decisions.
    - **External Authorization Policy Store** – provides policy related to authorizing access to OIM services.
    - **External Authentication Policy Store** – provides policy related to whether OIM will accept a set of evidence for authentication.
    - **External Authorization Policy Decision Point** – determines whether or not an access to OIM services should be authorized based upon a particular set of

attributes of that access attempt. OIM may simply apply this decision, or its internal PDP may integrate it with a higher level decision that considers additional attributes.

- **External Authentication Policy Decision Point** – determines whether OIM should accept a set of evidence for authentication based on a set of attributes of this authentication attempt. OIM may simply apply this decision, or integrate it with a higher level decision that considers additional attributes.

- **Computer Network Defense:** External functions that accept data from OIM and use it for computer network defense. There may be any number of these unique functions, with different information requirements.

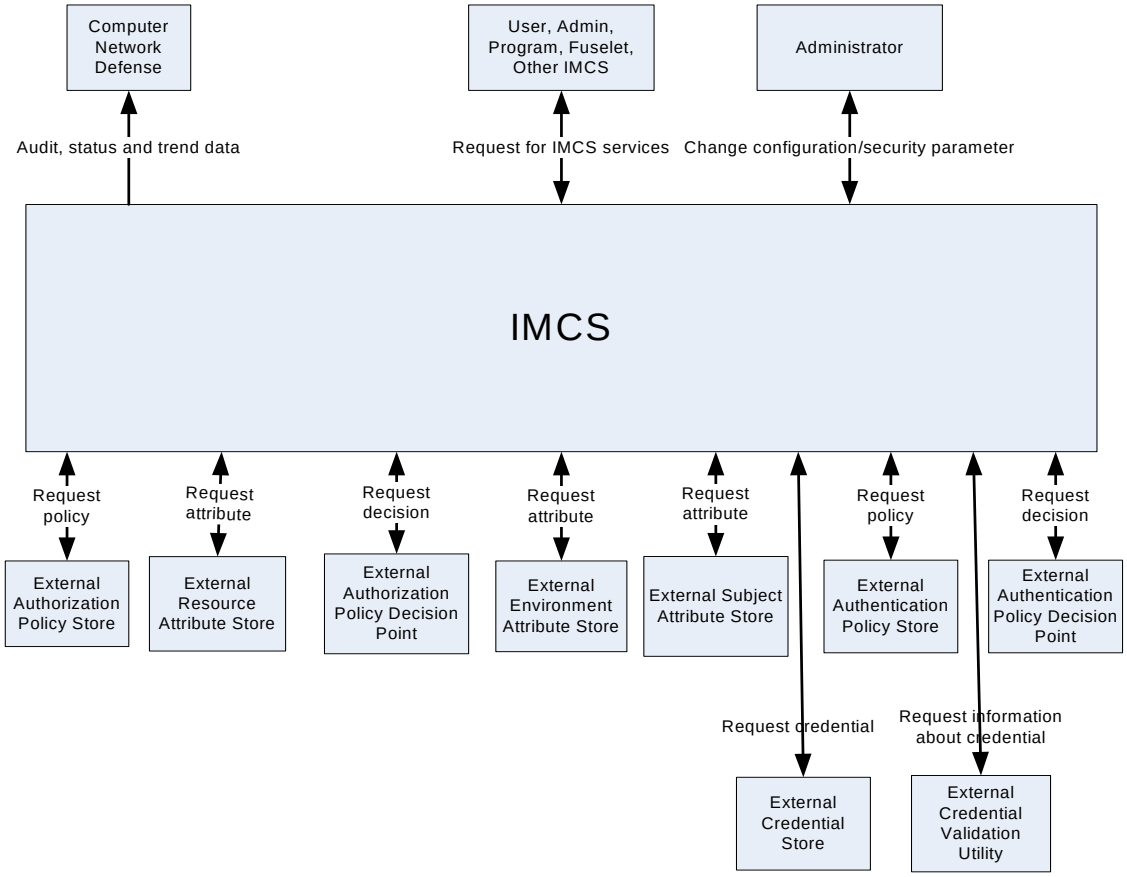

Figure 2 - OIM External Interfaces

Not shown in this figure are potential feeds from external systems that provide updates to OIM internal stores. Examples are synchronization functions with external stores of the same data, or live feeds of information such as INFOCON status.

It should be noted that the system is capable of operating with only the Client and Administrator external interfaces in place. The specific interfaces required by an application that uses the OIM

framework will depend upon the ownership and location of the infrastructure data upon which the security of that application depends.

6.3 Architecture Key Characteristics

In this section we summarize key characteristics of the architecture from the overview just presented, and highlight insights from the detailed analyses presented in the remaining sections of this report.

6.3.1 Basic Authentication and Authorization

The Policy Enforcement Point module is the single interface for authenticating and authorizing entities to OIM. It relies on Policy Decision Points for Authentication and Authorization, which in turn rely on modules that store and retrieve policies and the attributes upon which those policies are based. These attributes can be for subjects, resources, or the environment.

The architecture does not dictate an authentication protocol, although we show how both password and certificate based authentication fit into the architecture in Sections 9.2.2.1 and 9.2.2.3. The architecture supports any authentication algorithm that is realized as a protocol exchange between a client and the server that wants to authenticate them. Note that the architecture does not explicitly support authentication protocols that involve a third entity in the protocol other than OIM and the client, such as the token-granting server for Kerberos. The impact of such an authentication method on the architecture is identified for future study. GIG support for such a protocol, specifically an assured implementation of such a third party server, appears further out in time than those protocols we cover here. Section 7.1.1.1.2 contains further discussion of this topic.

The architecture allows for internal or external utilities that assist in credential validation for various protocols. These utilities may do all or any portion of this validation process.

Both authentication and authorization policies can be based on subject and environment attributes. Authorization policies typically also take into account resource attributes, such as metadata values. However, one cannot necessarily optimize retrieval from subject and environment stores to meet the needs of both functions. This is because in the session based environment, authentication may occur significantly separated in time from authorization, and attribute values could become stale by the time authorization takes place.

6.3.2 Authentication Assurance Level

The Authentication Policy Decision Point obtains the required data to feed a separate module that calculates authentication assurance level. This value can be used as input to either or both authentication and authorization decisions. It can be calculated whether or not OIM is accessed as a web service. Its operation within the authentication process and handoff to the authorization process is described in Sections 9.2.2.1 and 9.3.2.1.

6.3.3 Authenticating and Authorizing Automated Requesters

There is no intrinsic structural difference in handling of authentication and authorization for automated requesters vs. human requesters. The architecture illustrates automated entities authenticating with certificates, though this is not a limitation. Further, some advanced subject attributes and the concept of authentication assurance level are unlikely to have useful analogues for automated entities for some time. This simply means that the architecture has more functionality than needed for automated requesters. However, the system needs to be able to distinguish human from automated requesters, via data in their certificate. Section 9.2.2.2 discusses these topics in further detail.

6.3.4 Session vs. Transaction Model

OIM supports login sessions. It also supports individual transaction requests, both to operate as a web service and to service requests from federated OIM instances. Thus OIM has a Session Manager, a Message Manager and an Incoming Remote Request Handler module.

For authentication and authorization, the key differences between supporting login and transaction based interfaces are:

- Successful authentication of a transaction automatically kicks off authorization processing. The session model allows authentication (login) as a unique operation, and other requests are made and authorized some time after login occurs.
- Since login separates authentication from authorization, the system needs a mechanism to remember that someone is logged in. This is handled by the Session Manager, and no similar function is needed of a Message Manager.
- At completion of authorization, in the web service process flow, the PEP asks Information Services to respond to the client.
- Upon completion of the authorization process flow in the non-web services case, the PEP provides the client with a handle to allow them to interact with Information Services for the action requested.

Beyond these differences, the modules and processing flow involved in supporting session and transaction based authentication and authorization are the same. Sections 9.2.2.4 and 9.3.2.2 describe these flows for the web services transaction based case and compare them to the session-based case.

In particular, the use of authentication assurance level and flexible policy attributes for authentication and authorization policies can be integrated in a transaction-based NCES compatible environment.

6.3.5 Federation

One OIM instance becomes federated with another, when each defines authentication and authorization policies that either:

- Allow requests to be sent to the other
- Allow responses to be sent, to requests received from the other.

In addition, authorization policies may be defined which:

- Determine whether an OIM instance will accept authentications asserted in requests from federated OIM instances
- Allow transmission of specific client attributes to federation members in requests
- Mappings of these attributes to attributes meaningful to federated OIM instances
- Whether attributes asserted within a federated OIM request will be accepted

An OIM instance can maintain attribute mappings for its federated partners in a local policy store, or attribute mappings may be maintained for a federation in an external policy store, and retrieved from there by all federation members. This is the concept of an "Attribute Service" as defined under the WS-Federation standard (Section 7.1.4.3).

Federation processing relies on three modules for finding data in the federation, requesting it, and accepting such federated requests, respectively. Briefly, the federation model operates as follows:

- The requester only logs into one OIM instance via its PEP.
- This OIM instance handles finding any relevant data in other OIM instances.
- This OIM instance handles forwarding requester information, which may include their authentication information, and attributes used for authorization, and requests this data on the requester's behalf.
- An OIM receiving such a request authenticates and authorizes first the sending OIM, then the end requester.
- These authentication and authorization processes use the basic authentication and authorization modules.
- The receiving OIM may or may not honor the asserted end requester's authentication and attribute information.

The proposed federation architecture does not support "chaining" of requests, where a request is sent from OIM to OIM and then perhaps to another OIM, before it is finally fulfilled.

Since each OIM instance can control its authentication and authorization policies, it is possible that a federated request may be refused. xThis implies that the requester could get "partial

results" for a request where some relevant data resides on their local OIM and other relevant data resides on one or more federated OIM's.

OIM itself requires an identity to support federation and the server-to-client half of some authentication protocols. xA certificate and key store holds OIM's own private key and certificate.

Federation process flows are analyzed in Section 9.4.

6.3.6 NCES Compatibility

The requirements to send out requests to NCES compliant web services, such as NCES Security Services, include most the requirements for OIM itself to operate as a web service. This is because processing of the responses to requests that OIM sends to a web service involves almost all of the functions needed to process a web services request that is not a response, but rather originated from a client. The impact of this support on the architecture is:

- The Policy Enforcement Point manages different processes for the handoff from authentication to authorization in a session based and message based model
- The addition of a Message Manager module and a module to package outgoing NCES compliant messages
- Support for the specific authentication protocol that consists of receiving and processing NCES compliant messages according to NCES guidelines. xFrom the architecture point of view, this is not a structural impact, but rather support for an authentication protocol "exchange" that consists of one inbound message, where the credential is the web service request message itself.

This architecture does not utilize the NCES authorization architecture current as of this publication, as it is not applicable for OIM needs. See Section 7.1.2 for details on this topic.

6.3.7 Computer Network Defense

The authentication and authorization processes are an important source for audit records, statistics and trend data to support computer network defense (CND) processes. Since the Authentication and Authorization Decision Point communications of their decisions back to the PEP provides a common reference point in all processes we developed, this provides the point at which CND data is generated and sent to the various recipients that require it. We have integrated the most detail on these features into the process flow for Authorize Client Action in Section 9.3.2.1.

Although CND falls outside the immediate scope of authentication and authorization, the architecture also suggests a technique to remain flexible to provide this kind of data in virtually any format to as many recipients as may need it. This is done by separating data generation from formatting and forwarding functions.

6.4 Framework Options

The proposed architecture provides a variety of options in order to provide a framework suitable for a variety of applications.

Specifically, the architecture supports the following categories of options:

- Most authentication protocols
- Internal and external stores for attributes of subjects, resources and the environment
- Internal and external stores for policy
- Internal and external policy decision functions
- Internal and external certificate verification functions
- Web service or non-web-service interface from clients
- Federated or stand-alone mode
- Flexible set of policy parameters
- Use of, and algorithm for, authentication assurance level
- Configurable set of external targets for both audit and computer network defense data generated by OIM

The following subsections discuss each of these options. In particular we describe:

- What combinations of options are supported for use by a single instance of an application of OIM
- The general mechanism for providing the option in the context of this architecture
- Elements of the architecture that are optional.

6.4.1 Authentication Protocols

The architecture supports any authentication protocol that is implemented as a predictable message exchange between the client and server. However, to implement a new protocol, a submodule of the Authentication Policy Decision Point must be provided that implements that message exchange from the server side, and delivers a success or failure result from this exchange to the Policy Enforcement Point. A lower level architecture that realizes this flexibility is presented in Section **Error! Reference source not found.**. In addition, either the Policy Enforcement Point or the Authentication Policy Decision Point must be able to distinguish the particular authentication protocol that a client is initiating. The architecture does not dictate a specific mechanism for this.

A specific instance of OIM may simultaneously use any number of protocols to authenticate its users.

From purely the point of view of the architecture, an "authentication protocol" that consists of simply accepting a request from a user, would be valid. Although authentication itself is not optional in the architecture (it is always invoked), its realization can essentially make it so.

6.4.2 Internal and External Attribute Stores

The architecture supports OIM stores for attributes required for policy decisions, obtaining this data from external databases, or a combination of these. The following subsections discuss stores for subject, environment and resource attributes.

6.4.2.1 Subject Attribute Stores

A *subject* is a human, system, device or process that initiates OIM services. Basic attributes for subjects are distinct from the credentials possessed by those subjects. Examples of such attributes for human subjects are citizenship, identity assurance level, roles, COI membership, and active/suspended status as a GIG user. Specifically, we anticipate that:

- A GIG supported external user identity store for subjects that are human users, will become available, but may not be available for early OIM deployments, or cover all users or user attributes of interest to OIM. Other external or internal stores with smaller scope in terms of users covered, and potentially broader scope in terms of attributes stored, will remain in use, potentially in conjunction with the GIG supported capability. (See Section 11.0 for a description of GIG user attributes.)
- A GIG supported external store for attributes of non-human subjects, such as automated publishing or subscribing programs, will become available. However, the time frame is most likely further out than for support of such a store for human users. Thus for a significant time in the future, OIM would handle identity attributes for automated programs either internally or via other external stores smaller in scope than the GIG anticipates to provide.

Once usage of both external and internal stores is feasible, one must ask how a given instance of OIM determines how to fulfill a request for subject attribute information. The base framework provides a simple algorithm that requests attribute data internally and in external stores, where the order for searching the available stores is a configuration parameter, and the search continues down the list until the data is found or one reaches the end of the list. Stores that are authorized sources for this data are in fact part of the OIM authentication policy. This search algorithm covers cases in which there is no internal store or no external stores, or where one of these types of stores is augmenting the data in the other type. However, this retrieval algorithm is architected as a replaceable, pluggable component of the Subject Attribute Retrieval Function. This is because finer grained capabilities may be desirable for some applications, such as optimized lookups for specific attributes, or discovery and reconciliation of conflicting attribute values among stores. These more advanced algorithms may rely on more advanced policy information in internal or external authentication policy stores.

6.4.2.2 Environment Attribute Stores

We anticipate that a GIG supported environment attribute store will become available to cover globally tracked attributes such as DEFCON and INFOCON status, but may not be available for early OIM deployments. When it does become available, it is not expected to cover all environmental attributes of interest to OIM, such as local operating hours, INFOCON or computer network defense status specific to an OIM instance, or the environmental status of the workstations for OIM users. Other external or internal stores may include such attributes as these, and will remain in use, potentially in conjunction with the GIG supported capability.

The retrieval of environmental attributes is likely to be determined by the specific type of entity for which environmental attributes are desired, such as the GIG as a whole, military base, OIM instance, workstation. Thus the base OIM framework would provide a map from entity types to locations, in other words, a table that defines where to look up environmental attributes for each type of entity. As noted above in the case of subject attribute stores, this algorithm may not suit all applications and thus is architected as a replaceable, pluggable component of the Environment Attribute Retrieval Function.

6.4.2.3 Resource Attribute Stores

We anticipate that a GIG supported resource attribute store will become available. It seems unlikely that it will contain attributes keyed by the names of resources stored in OIM, as this is too fine a granularity for a GIG service. It is more likely to contain additional metadata fields, quality of protection and class of service attributes that are assigned based on metadata values held by OIM itself for its information objects. Such GIG capabilities are not anticipated in the short term. Thus usage of QoP and CoS in authorization decisions would mean these attributes are stored internally or in external stores with limited scope, for some time forward.

The types of attributes for OIM data resources that are stored externally are likely to be limited in number. Therefore the base framework provides a map from attribute names to locations, in other words, where to look up specific resource attributes. As suggested above for the cases of subject and environment attribute stores, this algorithm may not suit all applications and thus is architected as a replaceable, pluggable component of the Resource Attribute Retrieval Function.

6.4.3 Internal and External Stores for Policy

Understanding the full implications of supporting both internal and external stores for policy is among the most challenging aspects noted for this authorization architecture.

Following are the simplest options for how OIM could work, that correspond to the base algorithm given for user attributes above in section 6.4.2.1:

- Consult only an internal policy store
- Consult only an external policy store

- Consult an internal policy store. If a match is not found, consult an external policy store.
- Consult an external policy store. If a match is not found, consult an internal policy store.

Although the first two options appear very simple – they become more complicated if the choice of which store to consult is contingent on parameters about the decision being made. We call the algorithm for how to choose a store, a "choose authority" algorithm. For example, if one is examining a federated request from a COI member, one might consult an external policy store set up for the COI. Otherwise, consult an internal policy store.

Further, regarding the last two alternatives, note that in typical policy implementations, a "match" is always found, since there is normally a default behavior to implicitly cover everything not explicitly covered. So these two alternatives are only useful, if the policy technology used doesn't always produce a match.

A more complex possibility is:

- Consult one or more external policy stores and an internal policy store. Have an algorithm that determines how to reconcile the answers given by these policies.

For example, the external policy could consider a set of different attributes than the internal policy does. One policy might yield a "yes," and the other a "no." In some cases the external policy could be considered to override, and in others, the internal policy would. One would need to consider how flexible to make the "reconcile authorities" algorithm. An algorithm that made one or the other the overriding policy under certain conditions would be an example of a "reconcile authorities" algorithm.

Summarizing then, the system can:

- Consult a "choose authority" algorithm to determine what ultimate policy store or stores to consult
- Consult a "reconcile authorities" algorithm if more than one ultimate policy store is consulted and they give conflicting results.

From an architecture standpoint, this means that algorithmic pre-processing and post-processing must be taken into account when consulting policy stores. These algorithms could legitimately be considered a part of the policy themselves. A potential pitfall here is that if this flexibility is taken a step further, "choose authority" algorithms and "reconcile authorities" algorithms and associated data might themselves be managed internally or externally. Ultimately, this could lead to an infinite regression, applying higher level "choose authority" and "reconcile authorities" algorithms in order to locate the lower level "choose authority" and "reconcile authority" algorithms and data, and so on.

For this architecture, we assume that "choose authority" and "reconcile authorities" algorithms and data reside internal to OIM, and are replaceable, pluggable components. "Choose authority" algorithms are part of the Authentication and Authorization Policy Retrieval Functions, and

"Reconcile authorities" algorithms are part of the Authentication and Authorization PDP's. Data used by either of these algorithms resides in an internal Authentication or Authorization Policy Store. It remains of interest for future investigation to determine whether the complexity to support external provision of these algorithms can be justified based on examples germane to the users' missions.

6.4.4 Internal and External Policy Decision Functions

We anticipate that the intelligence to compute access decisions may be shared between OIM and external sources. In other words, we do not assume that the OIM Authentication or Authorization PDP is able to take into consideration all externally and internally maintained attributes of an access situation and render a decision based on these attributes. This means that some sub-decisions may be made externally to OIM, and then considered by OIM in making its decision. As an example, OIM might use a sub-decision from a COI policy decision point, to determine whether the COI will allow some access to a particular individual in that COI. OIM might apply this decision directly, or layer on additional considerations of current bandwidth available and priority of this COI's tasks. Thus in practice, a decision may be structured based on sub decisions which may be made external to OIM.

The selection and invocation of an external policy decision point for use in a specific decision is very similar to that described for internal and external policy stores in Section 6.4.3. Thus we expand the "choose authority" and "reconcile authorities" algorithms and data described in that section to include selection of an external PDP and integration of its results into the ultimate decision by the internal OIM PDP. Recall that "Choose authority" algorithms are part of the Authentication or Authorization Policy Retrieval Function, and "Reconcile authorities" algorithms are part of the Authentication or Authorization PDP. Data used by either of these algorithms resides in an internal Authentication or Authorization Policy Store.

This includes a very simple case in which all decisions are made by an external Authentication or Authorization PDP – however this is accomplished by configuring the OIM PDP to work this way, and not by removing the internal OIM PDP component. These components are required in the architecture.

6.4.5 Internal and External Certificate Verification Functions

We anticipate that an external GIG supported function for full certificate path verification, plus revocation checking for certificates, will become available. In the near term, either just revocation checking, or possibly neither of these capabilities, will be available via external GIG services. Thus the architecture provides that certificate validation can either:

- Be handled fully internally by OIM (both certificate path validation and revocation checking (performed by a Credential Validation Utility Function)
- Be handled fully externally (by an External Credential Validation Utility Function)

- Have revocation checking handled externally and certificate path validation handled internally (using both a Credential Validation Utility Function and an External Credential Validation Utility Function).

The determination of what portions of this validation are done internally or externally is made by the Credential Validation Function based on an authentication policy. Thus if a robust policy semantics is supported, it is possible to check some certificates internally and some externally, and leave out revocation checking on others, if this is desired. The simple default authentication policy set in the base platform would be to do revocation checking and certificate path checking internal to OIM. Likewise, once external capabilities are available, this policy could be changed to always do full external certificate validation. For this case, one should be able to install an OIM framework with no internal certificate validation components.

6.4.6 Web Service or Non-Web-Service Interface for Clients

We anticipate that for the foreseeable future, some clients will access OIM as a web service and some will not. One reason for this is that encryption requirements on clients to meet the NCES security architecture are not likely to be universally supported. The second reason is that web services does not support a login concept, but rather handles authentication, authorization and integrity for each transaction one at a time. This works well for occasional requests, but adds overhead that may be significant in some cases, such as an automated publisher or subscriber who executes a periodic update of many information objects to or from OIM.

Therefore in the proposed architecture, OIM simultaneously supports both web services and non-web services requests, where a login is provided in the later case. The architectural differences between the web service and non-web service case are summarized in Section 9.2.2.4.3. In particular, a transaction model for web services (vs. a login model) means that authorization processing always directly follows authentication. Thus the coordination done by the PEP for the web services case is distinctly different than for the non-web services case.

This architecture does not specify how OIM distinguishes web services requests from non-web services requests. This may be done using the port assigned for incoming requests, or by parsing characteristics of incoming requests.

6.4.7 Federated or Stand-Alone "Mode"

We call an OIM instance "federated" if it does either or both of the following:

- Accepts requests for services on behalf of subjects that have been authenticated by other OIM instances
- Sends requests to other OIM instances on behalf of users that it has authenticated.

In non-web-services mode, this means that OIM either:

- Logs in a user, and later sends a request to another OIM instance, on behalf of that user

- Accepts requests from another OIM instance, on behalf of a user logged into that OIM instance.

Whether an OIM can send requests or accept requests from a specific remote OIM instance is specified as part of its authorization policy. Thus sending or accepting a federated request to or from a specific OIM is a type of action to be checked against policy. In other words, the specific ways in which an OIM is "federated" are a matter of policy.

If an OIM will be set up not to accept incoming federated requests, it will not need its Incoming Remote Request Handler module. If it does not send outgoing federated requests, it will not need its Outgoing Remote Request Handler module. Thus the product should installable without either of both of these modules. This would create a true "stand alone mode," where change of policy alone could not allow federated activity.

When OIM responds as a web service, federation is accomplished by what NCES terms "service chaining." That concept is described in the NCES Security Architecture document and is being further developed by NCES. We do not cover this topic further in the present document. However, the approach taken here for non-web-services federation very closely resembles that taken by NCES. One difference to note is that the present architecture does not allow for the long request chains supported by NCES, where requests are passed along several hops to the final responder. There is only one "hop." The rationale for this difference is explained in Section 9.4.2.1.5.

OIM could be capable of simultaneously performing the following:

- Send out a request to a federated OIM in the form of a web service request
- Accept a request from a federated OIM as a web service request
- Send out a request to a federated OIM as a non-web-service request
- Accept a request from a federated OIM as a non-web-service request.

The mechanism via which OIM determines whether an outgoing federated request should be formatted as a web service request or not, is not specified by this architecture. As examples, it might be configuration information about its federated partners that is internal to the sending OIM, or may be information sent back by the discovery process that determines that a federated OIM has resources relevant to a local user's request.

6.4.8 Flexible Set of Policy Parameters

The architecture does not mandate a specific set of parameters that are considered in authentication or authorization policies. It does provide support for using arbitrary subject, environment and resource attributes as policy parameters, and for calculating authentication assurance level, which can be used in both kinds of policies. There are several levels of flexibility for policy parameters that one might support, listed here in order of increasing challenge for implementation:

1. **Compile time defined, evolvable parameter set:** All OIM instances (for a given OIM release) support the same set of policy parameters, even though each OIM instance may actually use just a subset of the full set. The maximal set is not limited by the architecture, so can evolve with each OIM release.
2. **Install time configurable parameter set:** A unique set of parameters can be defined at install time, but not necessarily modified after that point without a reinstall or upgrade.
3. **Parameter set modifiable without reinstall:** The parameter set can be modified on an installed system while the system is down for maintenance.
4. **Run-time modifiable parameter set:** The parameter set can be modified while the system is running.

The architecture proposed supports #1 at a minimum. It is a goal to support #4, and the KAoS authorization architecture does this (see Section 7.1.3).

Adding a policy parameter involves incorporating operational software that can:

- Construct policies that include the parameter
- If the parameter is to be stored internally, add values for the parameter to the appropriate internal store
- Determine the value of the parameter for policy evaluation.

Therefore, support for run-time modifiable policy parameters has the following implications in terms of this architecture:

- The Security Administration module must allow run-time pluggable submodules that add user interface and internal functionality for defining policies that consider new parameters.
- The Resource, Environment and Subject Attribute Stores and Retrieval Functions must allow run-time addition of new attribute types.

The default algorithm defined in Section 6.4.2.1 for retrieving an attribute was to search available stores in some order. This would not need modification when a new parameter was added.

The default mechanism defined in Section 6.4.2.2 for retrieving an environment attribute from the available environment attribute stores is to reference a table that defines where to look up environmental attributes for each type of entity. This mechanism supports run-time addition of environment parameters without change as long as the new parameter conforms to this default concept.

The default mechanism defined in Section 6.4.2.3 for retrieving resource attributes is to recognize locally stored attributes, and to find externally stored attributes using a map from attribute names to locations. This mechanism supports run-time addition of environment

parameters as long as the internal data allowing recognition of local attributes and the external attribute look-up table itself are run-time modifiable, at least for additions.

Note however, if any of these default algorithms were replaced for a particular OIM, one should consider the impact of the replacement on the ability to support run-time additions of attributes of the various types as policy parameters. A specific OIM application may find one of the flexibility levels #1-#3 acceptable, or may wish to retain the run-time flexibility.

6.4.9 Authentication Assurance Level Use and Algorithm

A specific OIM instance may not use authentication assurance level in its policies, and for this case the Authentication Policy Decision Point should be configurable such that it need not kickoff the calculation of this value nor pass it along to the Policy Enforcement Point.

It is anticipated that the GIG will define a reference calculation for authentication assurance level that would be implemented as a default in the base OIM. However, it is also anticipated that there will be some latitude in the use of this reference calculation. Therefore the minimum flexibility provided should be that the algorithm for authentication assurance level, based on a fixed set of inputs delivered to it by the Authentication Policy Decision Point, should be replaceable with another algorithm based on these same inputs, when the system is down for maintenance.

6.4.10 Targets and Formats for Audit and Situation Awareness Data

New requirements for reporting audit data and situation awareness data to external entities may arise at any time, and may be urgent based on a computer network defense situation. Therefore a run-time capability to modify the set of targets and the data sent to existing or new targets should be supported. In the proposed architecture, this is realized as run-time configurable parameters of the Data Filter and Store/Export module.

7.0 Rationale

7.1 Relationship to Other Architectures

7.1.1 Proposed GIG Increment 1 Architectures

This study found significant leverage in current GIG architecture work, to the extent that a number of modules defined in this architecture have been identified in GIG studies. Other OIM modules described here realize GIG concepts, for which the GIG work had not yet reached the module definition phase. Sections 7.3 and 7.4 summarize these impacts. In this section, we present our study of the GIG IA analysis of alternative architectures, specifically for identification and authentication (I&A) and for authorization, and how these alternatives relate to OIM and this architecture.

Note that in support of this architecture effort, we consulted GIG Information Assurance engineering outputs informally, in the form of a work in progress. Therefore our comparison and analysis below of GIG alternative architectures may not be valid for the final GIG Increment 1 architectures. The GIG draft references used this analysis were:

- Identification and Authentication Technical Report[7]
- GIG Policy Based Access Control (PBAC) Framework Technical Report[8]

These references provided architecture alternatives and comparisons of these alternatives along a number of dimensions, including security, efficiency and availability.

7.1.1.1 GIG Identification and Authentication

The GIG I&A Framework document describes three alternative architectures for I&A:

- Alternative 1: Service End to End, Authentication End to End
- Alternative 2: Direct Service Access, Authentication Through Intermediary
- Alternative 3: Service Through Intermediary, Authentication Through Intermediary

In the subsections below we describe these alternatives and compare them to the proposed OIM architecture.

---

[7] GIG Identity Management and Authentication Team, Identity Management and Authentication Framework Technical Report, Version .7, September 5, 2005.

[8] NSA IA Architecture Office (I11), GIG IA Independent Framework Technical Report for Policy Based Access Control, Version 1.0, September 8, 2005.

Note that the GIG analysis gives specific authentication protocols as examples, but the general analysis is at a higher architecture level.

7.1.1.1.1 Alternative 1: Service End to End, Authentication End to End

The process flow for GIG I&A Alternative 1 is:

- Consumer establishes a session to a service provider
- Service provider and consumer mutually authenticate each other
- Service provider requests consumer credential status from an external system as needed
- Service provider provides service directly to consumer

This OIM proposed architecture supports GIG I&A Alternative 1. Authentication with a CAC, a password or a thick client that supports the NCES security architecture are examples, as described in Sections 9.2.2.1, 9.2.2.3, and 9.2.2.4, respectively. The GIG Framework document sees Alternative 1 as most attractive overall.

7.1.1.1.2 Alternative 2: Direct Service Access, Authentication Through Intermediary

The process flow for GIG I&A architecture Alternative 2 is:

- An intermediary provides security tokens to consumer client and service provider
- Consumer establishes an association with the service provider. Mutual authentication is based on the intermediary-provided consumer and service provider security tokens.
- Service provider provides service directly to consumer

The main example for Alternative 2 is Kerberos (the RFC 4120 protocol). The operational advantages of Alternative 2 are that it allows a single authentication to provide sign on to many services, and that it allows a "local" password to be "traded in" for a credential that is accepted more widely, as long as the local password is known to the intermediary.

The proposed OIM architecture does not support GIG I&A Alternative 2. The advantages of Alternative 2 seem attractive, but from a practical standpoint do not provide a clear win for the OIM environment. First, the prerequisite for obtaining reduced sign on to many services (e.g. OIM plus others that a user needs) is that all of these services must support the intermediary-based protocol. Reduced sign on limited to a federated group of OIM installations is provided via a different method by this architecture, where in fact only one OIM login takes place (see Section 9.4). Reduced sign on that would provide access to OIM *and* other kinds of systems, depends upon the unknown of support by other systems for the intermediary based protocol. Secondly, note that setting up an intermediary to know local passwords, and setting up a relationship of the intermediary with the end service, are both required in order to use password "trade in." It seems more straightforward and no harder to set up an external database with the local passwords that the end service can reference, so that these are no longer "local."

Although such a database would be subject to attack, so is an intermediary. In fact, this is the main disadvantage to Alternative 2 cited in the GIG I&A framework document – the ability at this point to provide an intermediary that is resistant to attack. In addition, such an authentication intermediary is an external security component which does not currently exist. The GIG document does point out that robust Alternative 2 authentication intermediaries may provide part of the answer in the future to cross domain authentication. Therefore it will make sense to revisit the attractiveness of this alternative along with future work on OIM MLS issues.

7.1.1.1.3 Alternative 3: Service Through Intermediary, Authentication Through Intermediary

The process flow for GIG I&A architecture Alternative 3 is:

- Intermediary (such as web site) and consumer establish session and mutually authenticate each other
- Intermediary establishes a session with service provider
- Intermediary presents available services to consumer client. Consumer requests service.
- Service provider authenticates intermediary based on intermediary authentication information
- Service provider provides service to intermediary (Service provider relies on the intermediary to authenticate the consumer.)
- Intermediary provides service to consumer

The key example of Alternative 3 is an application that provides a web portal front end to the user, where the portal interacts with OIM, and the user interacts with the portal. This scenario seems extremely likely, since it concentrates the logic of interaction with OIM in the portal, while the user maintains a familiar browser interface. Therefore it is recommended that the OIM architecture support Alternative 3. This means in particular that authentication for OIM must be capable of recognizing two subjects related to a single request: the requesting subject (the portal) and the end user. Authorization logic then also requires taking both of these into account. Both of these features are available in this architecture as proposed because they are required for federation, in which an OIM instance acts as an intermediary to service a user logged into another OIM instance. Note that besides the federation and web portal applications, this two-tier authentication model is likely be important for taking requests from fuselets on behalf of particular users.

There are two specific cases where OIM would offer an Alternative 3 style of authentication:

1. As noted above, a user interacting with a web portal, that in turn interacts with OIM, using a non-web-service interface
2. A user interacting with an NCES thin client (such as a web portal), that in turn interacts with OIM using an NCES compatible web service interface.

We have not provided process flows for these authentication sequences, although we anticipate that the flows for cases 1 and 2 can be constructed by reusing sequences in the federation processes in Section 9.4 and the NCES Security Architecture[9] process flows, respectively.

Subcases of case 1 above should allow OIM to either accept the identity of the end user from the intermediary, or to authenticate the end user directly. It would not seem necessary to authenticate end users that are using highly secured portals, nor prudent to assume that data owners always want to trust the authentication process of a "client" portal.

Although it would appear to simplify matters to offer only the web-services based interaction and associated NCES security model for portals connecting to OIM, this is not recommended. Passing large data sets through an intermediary where every message requires a digital signature is likely to create a performance penalty for some time into the future.

Even though one OIM may serve as an intermediary to another OIM via the proposed federation architecture, this process flow differs from GIG IA Alternative 3 because the target OIM need not accept the authentication presented by the requesting OIM. It may re-authenticate the user. In other words, it does not necessarily rely on the originating OIM to authenticate the user. This situation no longer falls under Alternative 3, but rather is a variant of Alternative 1, since the OIM that is going to service the user, is also going to authenticate the user. We call this a "variant" because the original request for service did not come directly from the user, but rather came via the OIM which originally logged the user in.

7.1.1.2 GIG Policy Based Access Control

The GIG PBAC Framework Document describes three alternative architectures for PBAC:

- Alternative 1: Constituent Systems Provide Policy Decision Point (PDP) and Policy Enforcement Point (PEP)
- Alternative 2: PDP is Provided as an Enterprise Service
- Alternative 3: PDP and PEP are Provided as Enterprise Services

The GIG IA analysis notes that the scenario that would lead one to provide PEP and PDP functionality external to a system is when (1) the system itself is in a mobile environment and cannot be adequately protected, while (2) the GIG PDP and PEP services reside in a fixed environment with adequate protection. The other major factor differentiating the alternatives is preference in terms of system availability – for this factor the preference ordering is Alternative 1, Alternative 2, Alternative 3. The greater number of communication links required to perform policy decision and enforcement, the greater the risk of unavailability of this functionality.

---

[9] Net-Centric Enterprise Services (NCES) Security Core Enterprise Services (CES) Architecture, Version .5, December 29, 2004.

The GIG analysis also concludes that Alternative 1 scores lowest on ease of administration, since policy elements are likely to need synchronization with data in other parts of the GIG, and this is easiest to do centrally. An additional factor to consider, not discussed in the draft GIG document reviewed, is that a central policy administrator is less likely to be able to provide the flexibility and responsiveness to intricate operational needs that a local policy administrator might offer.

In summary, the proposed OIM architecture provides for Alternative 1, and provides a variant of Alternative 2 merged with Alternative 1. The GIG document states that Alternatives 1 and 2 cannot be offered by they same system simultaneously. However, we find that OIM needs to offer some capabilities of each of these options simultaneously; hence we designed a way to "merge" them, and thus offer a hybrid Alternative1/2, where PDP's both internal and external to OIM may be consulted for a policy decision.

Specifically, the point that Alternatives 1 and 2 cannot be used simultaneously is simply that either policy decisions are made externally, or internally, but cannot be made in two places. The proposed OIM architecture takes a broader view of what a PDP does. Specifically, a PDP may make intermediate decisions that are then used by another PDP to make the final decision that determines access. Thus there may be an external PDP that is consulted by an internal PDP, or vice versa. For example, a COI may have a PDP that consults attribute stores and its own COI maintained policies to make a decision as to whether one of its members can see a resource. The system that provides that resource might request and enforce the COI policy decision verbatim, or it might layer on top of it a policy that considered its own current threat environment, or the demand for bandwidth by other higher priority COI's. Likewise, in some cases, an external PDP might have the final word on access, after considering the output of a locally generated policy decision. Thus policy decisions can be constructed based on "subdecisions."

The GIG alternatives state that attribute data that goes into policy decisions can be data that is maintained either locally or externally. One could therefore theoretically fit these kind of scenarios into the GIG architecture model by calling the policy decision made by the COI PDP an "attribute," albeit an complex attribute that needs real time calculation based on a policy. However, we chose to highlight this complexity explicitly. One conclusion from this architecture study is that reconciliation of various levels of policy is one of the most challenging open issues for OIM access control, and GIG access control in general.

The proposed architecture does not provide for GIG PBAC Alternative 3. Moving the Policy Enforcement Point external to OIM the system is a greater threat to availability than moving the Policy Decision Point. This is because unavailability of the external PEP service means that OIM is unavailable – unless a way around the PEP is found, which would be a critical blow to the security of OIM. However, if an external PDP is unavailable, caching by the PEP is possible that can allow OIM operations to continue. If an OIM application was identified for a high risk mobile environment, where brief downtime periods were considered an acceptable tradeoff to lower the risk of successful attacks on the system, this decision should be revisited.

7.1.2 NCES

Under the proposed OIM architecture, OIM can serve clients as an NCES-compatible web service and is compatible with the NCES security architecture for authentication. The proposed OIM architecture does not conform to the current NCES authorization architecture, but rather is designed for attribute-based policies, which is an aspect of the future evolution identified for the NCES security architecture. Specifically, we see little utility for OIM of the service level granularity of policy that is the focus of the current NCES authorization architecture. This section will expand further on these comments.

For authentication, the current NCES architectures for thick clients and thin clients (such as users behind portals) are examples of GIG I&A alternatives 1 and 3, respectively, discussed in Section 7.1.1.1. Section 9.2.2.4 presents a detailed process flow showing OIM support of NCES thick clients, where OIM performs authentication in a manner compatible with the NCES security architecture description of this capability. The OIM architecture integrates GIG concepts with the NCES architecture beyond those detailed in the NCES security architecture document itself, in particular the use of authentication policies and authentication assurance level in conjunction with NCES compatible authentication using signed SOAP messages.

Section 7.1.1.1.3 outlined the recommendation that OIM support NCES thin clients such as web portals that access OIM as a web service. Details for these data flows do not appear in the present document, but from an authentication point of view are expected to follow the detailed diagrams in the NCES architecture document, using fragments from the federation process flows that appear in Section 9.4.2. These flows describe authentication of a remote federated OIM followed by acceptance of that OIM assertion of authentication for an end user. These flows can be reused because OIM processes for receiving (non-web services) requests from other federated OIM instances on behalf of clients logged into those remote systems, were in fact modeled after NCES processes for receiving web services requests from portals on behalf of NCES thin clients.

For authorization, the NCES architecture in its current form focuses on decision and enforcement of policies at the granularity of service operations offered by a web service. For OIM, this means, for example, a policy about who can publish. The NCES architecture visualizes that a web service will have an incoming SOAP message handler that enforces a service operation granularity policy. A scope statement in the NCES architecture document notes that enforcement of data level policies is assumed to be covered by applications. As we understand it, at the time of our review, the NCES authorization architecture requires structuring two layers of policy that are applied sequentially, where the first layer determines whether the subject can invoke the service operation based on a policy at that level, and the second layer then applies more detailed data level policies. This is two decisions and two calls out for policy data. The requirement to structure policy and enforcement to fit in a partial enforcement mechanism at the web services operation level has several disadvantages:

- It makes policy structure dependent upon the fact that a web services interface is being used. If possible, one would structure policy as flexibly as possible, to fit into arbitrary server implementations using future interface technologies. Further, for systems such as

OIM that support both web services and non-web services interfaces, this presents the challenge of maintaining and enforcing the same policy potentially cast in two different policy structures and enforced by two different architectures.

- A two level policy makes it difficult to understand the ultimate policy that will be applied. Would the system prevent the simultaneous presence of a service level policy that says a user cannot publish anything, and a data level policy that says that the user can publish type A and B documents? If so, how is this accomplished? If not, will the system be able to "compose" these policies in advance, to tell the administrator what would ultimately happen in this case – which is that the user would be blocked from publishing anything?
- Deny policies at the service level, such as a policy that says a particular user cannot publish anything, do seem useful. However, note that under the NCES authorization architecture, they would be enforced separately from data level access controls, using the SOAP message handler, using on a separate policy and at a different enforcement point-in-time than a policy that says a particular user can publish type A and type B documents. Therefore there is opportunity for conflicts and confusion about what effective policy is ultimately being applied. If deny policies at the service level are useful, they should be implemented as part of the same enforcement and policy infrastructure that is implementing data level access controls.
- In and of themselves, "allow" policies for OIM service operations are not expected to be very useful. An example would be – a particular user can subscribe to anything, or can publish anything. This is a rare case. And if one needs to specify which specific types of information objects a user can publish, there is little utility in first applying a policy that says that the user can publish.

In summary, if one views the "action" or service associated with an access control decision as simply one parameter associated with the attempted access, on par with all other parameters such as environment, user, and so on, it is not recommended to break off "action" with separate treatment for creation of policies and enforcement. The concept of attribute based enforcement is that logical combinations of attributes drive policy decisions, not individual attributes or even pairs of attributes.

NCES allows, but does not require, use of the NCES Policy Decision or Policy Retrieval services. Not using the NCES Policy Decision Service corresponds to GIG PBAC Alternative 1 (use of internal PDP). Using this service corresponds to GIG PBAC Alternative 2 (use of external PDP). The proposed OIM architecture supports PBAC Alternative 1 (external PDP) in addition to a hybrid Alternative 1/2 which includes both an internal and an external PDP (see Section 7.1.1.2). NCES does not currently identify an enterprise policy enforcement service, thus does not have an option that is an example of GIG PBAC alternative 3. Neither does the proposed OIM architecture support GIG PBAC Alternative 3.

OIM can take advantage of NCES security services as they become available, although it does not require them in order to operate. Specifically, Table 8 maps NCES services mentioned in the

NCES architecture document to external interfaces identified by the OIM architecture in Section 6.2. The NCES services noted as "future" in the NCES Architecture document referenced here are marked with an asterisk below.

Table 8 - Map of NCES Services to OIM External Interfaces

NCES Current or Future Service	External Entity Identified in OIM Authentication and Authorization Architecture
Policy Decision Service	External Authentication Policy Decision Point External Authorization Policy Decision Point
Policy Retrieval Service	External Authentication Policy Store External Authorization Policy Store
Policy Administration Service	*None – would be addressed once administration is covered by OIM architecture*
Certificate Validation Service	External Credential Validation Utility
Certificate Retrieval Service	External Credential Store
Principal Attribute Service	External Subject Attribute Store
Policy Subscription Service*	*None – would be addressed once options for synchronizing external and internal policy stores are detailed*
Certificate Registration Service*	*None – used at initialization to get OIM's own certificate - would be addressed once secure initialization is covered by the OIM architecture*
Resource Attribute Service*	External Resource Attribute Store
Environment Attribute Service*	External Environment Attribute Store
Domain Inquiry Service*	*Further study needed to determine utility for OIM*
Security Context Service*	*Further study needed to determine utility for OIM*
Secure Logging Service*	*Out of scope for this study*
Auditing Service*	*Not addressed – however, this could be an alternative to the internal Audit Data Generator, Situation Awareness Data Generator and Data Filter and/or Store/Export Function proposed in this study*

7.1.3 KAoS

KAoS[10]is an architecture and research implementation of advanced policy control for agents. Attractive characteristics of KAoS are:

- Ontology based, meaning that known relationships among entities in the environment can be leveraged in policy development
- Run-time changeable and extensible ontologies and policies
- Policy conflict resolution
- Generic framework to gather data at time of policy evaluation.

OIM is executing a program to examine how KAoS might be useful, and early NSA studies[11] have identified KAoS as a candidate technology to support RAdAC (Risk Adaptable Access Control), which is the future vision for GIG access control. RAdAC supports policies that weigh operational need against security risk.

The KAoS architecture breaks access control into Directory Service, Guard and Enforcer modules, which parallels directly the Attribute Stores, Policy Decision Point and Policy Enforcement Point modules in the proposed OIM architecture, and in the GIG IA architecture work to date.

At first glance, the apparently most striking difference between this OIM architecture and KAoS at the process flow level is that in KAoS, the Guard caches all policies relevant to the agents it controls. Thus when a new or changed policy is committed to the KAoS Directory Service, it is automatically distributed or "pushed" to all Guards that might use this update. This mode of operation allows the Guards to operate at full functionality in the event of loss of communication with the Directory Service.

Strictly following the above module correspondence between OIM and KAoS, the equivalent of this concept for OIM would be – the Policy Decision Point would cache all policies it might reference, and if a policy changed in any internal or external policy store referenced by OIM, then the Policy Decision Point would add this immediately to its cache. The proposed OIM architecture does not operate this way. It is not practical because OIM is likely to reference large

---

[10] A. Uszok, J. Bradshaw, R. Jeffers et.al., KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction and Enforcement, Institute for Human and Machine Cognition, Proceedings of Policy 2003, Como Italy. One of collection of KAoS papers at http://www.ihmc.us/research/projects/KAoS/Publications.php

[11] National Security Agency Information Assurance Directorate, IA Architecture Office (I11), RAdAC Feasibility and Possible Realizations, Revision 1.0, March 5, 2005

external stores that do not "know" the data of which an OIM instance needs to be made aware. The PDP simply asks for a policy when it needs it, or uses "pull." In the OIM environment, there is typically a policy store co-located with the Policy Decision Point, so that loss of communication no longer drives caching all policies in the PDP module itself. It is possible that an OIM might reference externally maintained policies only. In this case, if there was a possibility for loss of communication with these external policy stores, we would expect OIM to implement a local policy store that was kept in synch with the required external stores. Therefore this difference from KAoS disappears if one considers internal policy stores to part of the OIM modules that "correspond" to the KAoS Guard.

A more subtle distinction that arises when one looks more closely at what happens when KAoS distributes a policy to the Guards. Policies in KAoS are expressed in OWL, and the Stanford Java Theorem Prover is applied to reason about the KAoS ontology in order to evaluate policies. However, this analysis is not done at policy decision time – it is done in advance at the Directory Services before distribution of policy to the Guards. Thus Directory Services "pre-compiles" the policy into a form that can be applied by the Guard using the more typical mechanical policy evaluation method of "go down the list until you get a hit." This precompiled version includes ontology information as needed to apply the policy. The equivalent of this notion for OIM would be that the policies are stored in Policy Stores, in a format distinct from that used by the PDP to make decisions. Retrieval by the PDP of policies from the Policy Stores would include a pre-compilation step to render the policies suitable for enforcement. This could be done by the Policy Retrieval Functions in the OIM architecture. It is not recommended that OIM count on such a pre-compilation being done by external data stores, parallel to KAoS directory services, for the following reasons:

- capabilities of the PDP's that an external policy store might service will vary (and some may not be OIM systems), making this store the wrong place to put such functionality
- NCES favors communication of policies in XACML format. Although more advanced policy languages are likely to follow, this standard is likely to drive how GIG-provided external policy stores interface with their customers for some time.

Thus the functionality to pre-compile policies before enforcement could be supported by the proposed architecture, but lies below the level of the detail of the architecture description in this document.

The typical policy evaluation architecture collects all attributes used by policy and then looks for a matching policy. Our study of KAoS also pointed out that a consequence of a flexible and extensible policy is that this architecture becomes inefficient – because one not only needs to examine all policies to look at all attributes that need to be collected, but needs to collect them all up front– and many will ultimately not be needed. KAoS addresses this issue by having the PDP store and traverse the whole policy, collecting attributes as it needs them, to find a match to a policy situation. We have refined this solution to have the OIM PDP obtain a subset of the entire policy to traverse, based on key policy parameters. Section 9.3.2.1.3 provides further discussion of this point.

KAoS also points out the need and demonstrates the feasibility of providing run-time changeable and extensible policies. The impacts of this capability fall a level below this architecture description, but are discussed in general in Section 6.4.8.

7.1.4 Commercial Federation Architectures

In this section we describe how the proposed federation architecture described in Section 9.4 is related to leading commercial architectures for federation. The objective of this investigation is to determine if the requirements that drive the commercial architecture, and thus the architectural concepts designed to meet these needs, are useful for the OIM security architecture. While there is some overlap between commercial needs and OIM environment needs, there are also significant driving commercial needs for which we see no parallel in the OIM environment. These conclusions are detailed below in section 7.1.4.3 and 7.1.4.4.

Note that we examine the commercial architectures at a functional and data flow level. Particular characteristics of their intended application environment, such as web-services or even simply being web-enabled, are below the level of technology detail presented in this OIM architecture proposal.

Specifically we examine concepts from:

- Liberty Alliance
- WS-Federation.

The following two sections overview these two standards efforts.

7.1.4.1 Liberty Alliance Overview

The Liberty Alliance Project, created in 2001, is an alliance of more than 150 companies, non-profit and government organizations from around the globe. The consortium is committed to developing an open standard for federated network identityy that supports all current and emerging network devices. Current management board members are AOL, Ericsson, Fidelity Investments, France Telecom, GM, HP, IBM, Intel, Nokia, Novell, Oracle, RSA Security, Sun, VeriSign, and Vodafone. Microsoft is absent from this initiative, as it is leading the WS-Federation initiative described in Section 7.1.4.2.

Liberty Alliance identifies network identity as the sum total of all identifying and attribute information for a user on the internet. This includes data provided to any number of service providers. Liberty views their key issue as connecting the various identity “islands” to provide better service to the user.

Key concepts are principal (the user), Identity Provider (or IdP, entity who authenticates principal), and Service Provider (or SP, entity who provides a service). “Federation” is defined as the establishment of a relationship between any number of IdP’s and SP’s.

Current standards from the Liberty Alliance are:

- ID-FF 1.2 (FINAL), the Identity Federation Framework
- ID-WSF 1.1 (FINAL), the Identity Web Services Framework
- ID-WSF 2.0 (DRAFT), the Identity Web Services Framework, Draft Release 2
- ID-WSF DST 2.0 (FINAL), the Data Services Template
- ID-SIS, a collection of Identity Services Interface Specifications

7.1.4.2 WS-Federation Overview

Five technology vendors – IBM, Microsoft, RSA, VeriSign and BEA – published a technology white paper in 2003 titled, "Web Services Federation Language (WS-Federation)." It can be found at:

http://msdn.microsoft.com/webservices/webservices/understanding/advancedwebservices/default.aspx?pull=/library/en-us/dnglobspec/html/ws-federation.asp#ws-federation_active .

The WS-Federation standard draft states that its primary goal is to "enable federation of identity, attribute, authentication, and authorization information."

WS-Federation relies on other OASIS standards including WS-Security.

Key members involved in federated identity specification and standards - OASIS, Liberty Alliance and the IBM/Microsoft led WS-* --met at Digital ID World 2005 in May 2005 to discuss their different approaches to deriving specifications and the possibility of convergence. There is no agreement on this, but both groups appear more willing to leverage results from the other group than in the recent past.

7.1.4.3 Commercial Federation Features vs. OIM

The table below describes key business issues addressed by the Liberty Alliance and WS-Federation standards, and their relevance to OIM.

Table 9 - Relevance of Commercial Federation Issues to OIM Environment

Commercial Business Issue	How Addressed by Liberty Alliance and WS-Security	Relevance to OIM Architecture
Different user ID for each commercial web site	Liberty ID-FF *federated name identifier*, can be agreed well-known name, transient name (for anonymity) or pseudonym. ID-FF *Name Identifier*	The OIM environment, will use a well known name, the GIG user name. This is not subject to a negotiation as in the commercial case. Changes to GIG user name

	Mapping supports maintenance of user names across federation. ID-FF *Management Protocol* supports request by SP to IdP, asking for name known to another SP. WS-Federation *Identity Provider/Security Token Service*	do not originate with end services. Changes to GIG user names are made via GIG identity services. A user may have several GIG user names, but these correspond to different roles and privileges and thus are not mapped to each other during a login session. The user must use the appropriate name for a given action.
Different names for same attribute across partner organizations	Liberty: unknown WS-Federation *Attribute Service* use of *identity mapping*	In the OIM environment, an option will exist to provide mappings the attribute names, as noted in Section 6.3.5. For web services, the DoD Meta Data repository is supposed to assist in the deconfliction process.
Anonymity of user to partner organizations	Liberty ID-FF *anonymous* user, *pseudonyms*. WS-Federation *Pseudonym Service*	In the OIM environment, user anonymity is not a requirement. User accountability is a requirement.
Privacy of personal attributes provided to one organization, from others	Liberty ID-WSF *permission based attribute sharing* WS-Federation *policy* for *Attribute Service*	In the OIM environment, end users do not determine who can see their attributes. This is determined by higher level policy. The policies regarding personal information follow the guidelines set forth by the Privacy Act guidelines established by DoD 5400.11-R. In theory, this basic concept of restricted attribute sharing also applies to within the DoD as different organizations are not willing to “share” or even

		allow outsiders to know, via attributes, what information they do have. This is particularly applicable to the Coalition environment. An OIM instance may determine which attributes it is willing to share as described in Section 6.3.5.
Reauthentication needed when transferring to a partner web site, user must maintain many passwords. Need single-sign-on solution.	Liberty ID-FF uses concept of authentication *assertion*, either pushed out or requested from Identity Provider by target system, to provide "simplified sign on." WS-Federation uses concept of token (which is defined as "set of claims"), which is obtained by client from Identity Provider or Security Token Service, and then presented by the client to the target service.	Both pushed and requested authentications used in OIM architecture. Push case is used when target OIM for federated request accepts authentication from source OIM. Requested authentication is used when target does not accept authentication from source. Section 9.4.2 details the case in which OIM accepts the presented authentication evidence. For more details on the relationship between the proposed OIM architecture and the commercial concepts, see Section 7.1.4.4.
Some service providers may only accept certain kinds of authentications	Liberty ID-FF *authentication context* included in assertion. Recipient can base their acceptance of asserted authentication on this. WS-Federation – no support	Adapted the concept for OIM, authentication assurance level is included in assertion sent to target OIM for a federated request. OIM receiving federated request may apply an authentication policy using this or other attributes of the authentication event.

	as of 2003 Liberty comparison whitepaper referenced below Table 10.	
Service provider needs stronger authentication than asserted by requestor	Liberty ID-FF "up-authentication," using ability to request a specific authentication context when requesting an authentication assertion. WS-Federation - unknown	GIG includes future concept that reauthentication may be required do specific kinds of actions. Reauthentication for actions performed via a federated request may require interactions such as this. Details of OIM reauthentication processing to be defined by future efforts.
Coordinate logoff among federated entities	Liberty ID-FF *Single Logout Protocol* WS-Federation federated *sign-out*	OIM architecture does not use concept of login to federated systems. However, it does have parallel concept of Session Authority, which is the OIM PEP which logged in the user. This entity will manage all federated requests in-process when a user logs out. Details of this processing are to be defined in future efforts.
Provide specific service based on user identity, e.g. wallet, contact information.	Liberty ID-WSF *Identity-Based Services* and ID-SIS *Service Interface Specifications*	No requirements identified at this time for personalization of services based on GIG user, and discovery of these services.

7.1.4.4 Summary – Federation in the Commercial Business Environment vs. the OIM Environment

Based on the analysis in the previous section, the following table briefly compares driving requirements for commercial vs. OIM environment federation architectures.

Table 10 - Commercial vs. OIM Federation Drivers

Drivers - Commercial Business Environment	Drivers - OIM Environment
Users may need dozens to hundreds of identities for different uses if separate logins are required.	GIG environment will provide in most cases one GIG user ID per user. A few exceptional users will have a few GIG user IDs.
Various organizations will spend resources managing duplicate copies of user identities if separate logins are required.	GIG has same problems. Solution includes formalized and chartered GIG Identity Management functions that will manage GIG IDs. A single-sign capability must also be provided for user convenience.
Users have a right to privacy of their identity and attributes when they have not explicitly disclosed them to an organization.	End GIG users do not have a right to privacy for GIG transactions. Disclosure of attributes across organizations is subject to higher level policy, although support for such a policy has not been identified as a driving requirement by the GIG IA effort. The privacy policy is prescribed by DoD 5400.11-R. Individual OIM's may also have local policies about attributes they will disclose.
Federation agreement focuses on how users and types of authentication will be identified, and what attributes can be shared. Each individual organization determines what privileges each user will have, once they have gained access to that organization's systems.	Users will be identified by GIG ID. Federation agreement between GIG identity managers focuses on creating consistency in values of authentication assurance level. Creating a federation of systems to support a COI will consist of a joint agreement on how to set authorization policy so that COI members have appropriate access to the data they need. Appropriate access is ultimately defined by the leadership of the COI, not the owners of the systems that contain the relevant data.
Each federated organization has the right to maintain a unique architecture for authentication, as well as their autonomy in choice of operating systems, network protocols and databases.	Since all OIM systems will meet an OIM specification, there is the opportunity to simplify the design of federation by imposing an appropriate level of uniformity across OIM systems.

In 2003, the Liberty Alliance published a whitepaper comparing Liberty Alliance to WS-Federation (https://www.projectliberty.org/resources/whitepapers/wsfed-liberty-overview-10-13-

03.pdf). The relevant observation in that paper here, relates to the fact that the proposed OIM architecture uses "back-channel" single sign-on. A key difference pointed out in that comparison document is that WS-Federation recommends "front-channel" single sign-on, where a token (set of claims) is obtained by a client from an identity provider and then presented by the client to the target system. Liberty Alliance provides for both front-channel and back-channel single sign-on, citing the ease of use provided by the back-channel method. In the back-channel case, the Identity Provider communicates directly with the target service, rather than routing tokens back through the client, to have the client then present them to the service.

The reasons for selecting back-channel single sign-on for OIM are:

- Placing complexity in the server rather than client provides more development and assurance leverage, since it is anticipated that there will be fewer unique server software versions than client software versions.
- Placing complexity in the server rather than client places less demand on processing power and bandwidth out to client, potentially supporting a broader range of capabilities for client systems.
- For OIM, a front-channel single sign-on would be further complicated by the fact that the client does not know at the outset, which (if any) remote OIM instances need to be contacted to fulfill their request. Thus a protocol would need to be constructed in which the local OIM gives the client this discovery information, and then the client itself manages the requests to all of these remote OIM instances. This again places a large burden on the client.
- Back-channel method uses less communication "hops," with correspondingly less chance for network problems.

In summary, the requirement to avoid multiple user logins across several federated systems is the key area of overlap where commercial federation concepts find leverage in this OIM architecture. The architecture proposed here for single sign-on in a federated environment where the original authentication is accepted, copies "passive back-channel single-sign-on" as described by the Liberty Alliance. "Passive" means that an Identity Provider sends an authentication assertion to a service provider, where the service provider has not previously requested it.

7.1.4.5 Commercial Implementations

There are a number of commercial products that currently implement the "Federated" concepts. A few of these systems are described below.

7.1.4.5.1 Microsoft Passport

Passport is a relatively new service offered by Microsoft. This service allows a user to register at a central location, establish a profile, and "wallet" information. The user information typically includes an email address and password. The user profile includes personal data such as address,

phone number, etc. and the wallet, for commercial purposes is information such as credit card numbers. Passport assigns each user account with a 64-bit unique user id. When visiting a third party, Passport enabled website, the user is redirected to the Passport site for authentication. The authentication is stored in cookies and includes information from the Passport site, the third party site, and the local web browser.

Passport itself it not applicable to OIM as it is a centralized commercial service offered by Microsoft. Liberty Alliance standards offer an open standard for building a similar third party Identity Provider.

While the Passport is in use and demonstrates a number of key federation features, it does not appear to be directly applicable to the OIM. Primarily, this is because the redirection for authentication can produce unwarranted increases in bandwidth usage, unless there are multiple paths (e.g., many different servers in dispersed locations) that support the authentication process.

In the GIG/OIM environment, user attributes may be provided in several central locations on the network (and cached locally as needed), however, authentication is performed by OIM itself, perhaps with assistance from external utilities such as NCES certificate verification. In our federation architecture, one OIM may accept the result of another OIM's authentication. Thus in a sense, all OIM systems can act as identity providers to other OIM instances.

7.1.4.5.2 Kerberos

Kerberos is a single sign-on and authentication protocol that has been in existence for over 20 years. Kerberos is embedded in several commercial offerings including serving as the default authentication protocols on Windows. It is also supported by Solaris, MacOS, and Linux platforms. A Kerberos setup usually consists of client, server, and trusted third party mediator. The client logs on using a password which is turned into an encryption key by the server. The key is compared to the user's key stored in a database at the trusted third party. If the keys match, then the third party issues a ticket-granting ticket. The client sends the ticket granting ticket to the trusted third party when the client wants to connect to the server. The trusted third party returns a session ticket to the client to initiate the session.

The proposed architecture does not support Kerberos. See Section 7.1.1.1.2 for the rationale related to this decision.

7.1.4.5.3 e-Trust Identity Manager

Computer Associates offers the e-Trust Identity Manager, which was formerly a part of the Netegrity line of products. Geared towards the commercial market, this system provides the majority of the functions described in the commercial comparison. The system also offers a full auditing capability and management consoles for implementing policies.

7.1.4.5.4 Microsoft Active Directory Federated Service

Microsoft offers an upgrade to the Active Directory server that allows for a "Federated" implementation. This system also provides the functionality described in the commercial comparison. For the Air Force, this may be useful system in that it is an upgrade to currently fielded equipment.

7.2 Rationale from Process Flow Analysis

In this section we summarize elements of the module and interface structure of the architecture that were driven by insights from the construction of the process flows in Section **Error! Reference source not found.**. References are to the relevant process flow appendix section.

- **Session Manager and Message Manager:** A Session Manager is needed to support the non-web services "login" model. A Message manager is needed to support a message-oriented web services mode. (Section 9.2.2.4.3.)
- **Modules for Federation:** Interactions with other members of a federation requires functionality for data discovery and secure requests, handled by the Data Discovery module and the Outgoing and Incoming Remote Request Handlers. (Section 9.4.2.)
- **Interfaces for environment and subject attributes from both authentication and authorization PDP's:** In constructing the process flows, it was noted that attributes referenced at authentication time may need to be referenced again at authorization time, as their values may have become stale. (Section 9.3.2.1.3.)
- **Interfaces from both Authentication PDP and Credential Validation Function to Authentication Policy Retrieval Function:** The architecture supports policy driven authentication at two levels. First, the decision of whether to accept an authentication result can be driven by attributes in addition to the raw authentication protocol result. At a lower level, the operation of the authentication protocol itself may be impacted by policy, for example one may determine that in some situations a revocation check is not required to validate a certificate. (Section 9.2.2.1.4.)
- **Single interface to Authentication Assurance Level Calculator:** At first glance, a natural and self-contained design for the AAL Calculator would be for this module to obtain the data it needs for its algorithm from the various OIM modules. This design was ultimately rejected in favor of a single interface from the Authentication PDP to the AAL Calculator which carried all necessary data, because this module needs both the same data as the AAL Calculator and also an interface to the AAL Calculator. (Section 9.2.2.1.4.)

7.3 Review of Architecture Drivers and Decisions

In this section we summarize how the architecture achieves the drivers and implements the decisions identified in Section 4.0.

- **Meet and exceed GIG Increment 1 requirements:** GIG concepts in Increment 1 are realized as follows in this architecture:
    - Architectural separation of identity and credential infrastructures, achieved via separate Subject Attribute and Credential stores
    - Separate architectural elements for policy decision and enforcement, realized as Policy Enforcement Point and Policy Decision Point modules for both Authentication and Authorization
    - Separate architectural elements for retrieval and storage of environment, subject and resource attributes as defined by the GIG are used in this architecture
    - Support for external GIG authorities that supply subject and credential data and credential validation, by allowing for external interfaces to these modules
    - Support for authentication policy via the Authentication Policy Decision Point, the Authentication Policy Retrieval Function and the Authentication Policy Store.
    - Support use of certificates and passwords for authentication as described later in this section.

  Support for these advanced GIG capabilities is achieved as follows:
    - The architecture supports authorization policies based on **rich and extensible sets of policy parameters** by:
        - Changing the common sequential "gather data, then make decision" process flow to be an iterative process
        - Including functionality for reconciling internal and external attributes and policies in the Authentication and Authorization Decision Points.
    - Section 9.2.2.2 shows that authentication and authorization for **automated processes as clients** can use a process flow identical to that for human users, providing these clients are identified as such in their digital certificates.
    - The architecture includes an **Authentication Assurance Level Calculator** and describes its operation within the process flows in Section **Error! Reference source not found.**.
- **Use NCES Security Services or internal OIM security functions:** As noted by examining Table 8 - Map of NCES Services to OIM External Interfaces, the architecture supports external interfaces for all current NCES security services except for the Policy Administration Service, and supports some future services. The possible application of the Policy Administration Service would be studied in an extension of this architecture effort to cover the area of security administration. To identify the name of the internal OIM module that provides an alternative or complement to the external NCES service, remove the term "External" from the module names in the right hand column of Table 8.

- **Provide certificate and password authentication, while allowing a path for support of other methods:** The architecture is built around a generic concept of "credential" and a generic definition of an authentication protocol as a defined message exchange between an OIM client and an OIM instance. The message exchange and a credential validation must both be successful for the authentication protocol to succeed. The use of this model for certificates is described in Section 9.2.2.1, for passwords in Section 9.2.2.3 and for NCES compliant SOAP requests in Section 9.2.2.4. A proposed authentication architecture at the next layer down is presented in Section **Error! Reference source not found.**.
- **Support locally maintained policies and attributes, externally maintained policies and attributes, or a combination of both:** The architecture includes internal and external authentication policy, authorization policy, resource attribute, environment attribute and subject attribute stores. The process flows allow that either or both internal and external stores may be consulted for a transaction, under the direction of the associated retrieval function for this type of policy or attribute. Sections 6.4.2 and 6.4.3 describe how the retrieval functions could provide this functionality.
- **Provide transparent and secure access to data residing on any OIM instance (a.k.a. federation):** The federation architecture described in Sections 6.3.5, 6.4.7 and 9.4 allows a user logged in to any OIM instance to access information objects in other federated OIM instances. The "home" OIM locates this remote information using an OIM Discovery Module and mediates its return to the user, thus making information location transparent to the OIM client. Allowable interactions between federation members are controlled by policy at points of send and receive by the Policy Enforcement Point. Requests from federation members are authenticated by the receiving member to the originating OIM instance and may obtain an assertion for the original authentication of the end user, or reauthenticate them.
- **Integrate advanced support for Computer Network Defense:** The architecture includes an Audit Data Generator and a Situation Awareness Data Generator that both obtain data characterizing security decisions made by the system from the Authentication Policy Decision Point and the Authorization Policy Decision Point. These two Generator modules feed a Data Filter and Store/Export module that filters the raw data from these sources, formats it and stores it or transmits it to internal or external destinations.
- **Operate both in web service and API mode:** Support for web service mode required addition of a Message Manager module and the Web Service Outgoing Message Handler to package responses to clients in NCES compliant format. By treating incoming web service requests as a type of credential, the elements of the architecture covering authentication and authorization use the same structure to support both web service and API mode.

7.4 Rationale Summary

In this section we summarize the rationale for this architecture and its several sources, as detailed in the previous sections. The rationale derived from the following sources and interactions among these:

- Architecture drivers and decisions, where the impact of these on the architecture is described in Section 7.3
- GIG and NCES architecture studies, described in Sections 7.1.1 and 7.1.2, and summarized below
- Examination of the KAoS flexible policy architecture. KAoS provided a model for our process flow for evaluating rich policies, as noted in Section 7.1.3
- Consideration of commercial federation architectures, as described in 7.1.4. A number of issues treated in these architectures do not apply to the GIG environment (see summary in Table 9), but concepts for single sign-on, variable authentication policies across a federation and attribute mappings do apply.
- Insights from constructing the process flows in Section **Error! Reference source not found.**, particularly related to the interfaces required between OIM modules, as summarized in Section 7.2.

Figure 3 provides a map to describe the rationale for the overall set of modules in the OIM architecture. The figure highlights the modules that were identified in the GIG and NCES architectures.

Modules in the following areas of functionality do not come directly from GIG or NCES:

- Modules in right half of Figure 3 that implement authentication. Although the GIG IA effort has included authentication, architecture work has focused at a higher level on comparing general types of authentication protocols, rather than detailing an architecture with flexible support for a variety of protocols. However, the OIM module structure provides a model for architectural support of the GIG concepts authentication assurance level and authentication policy.
- Three modules used for federation operations, top center of Figure 3 which have not yet been treated at this level of detail by the GIG and NCES efforts. The discovery function and functions to request and accept results were located in the OIM server for assurance and transparency reasons as described in Section 7.1.4.4.
- Three modules for Computer Network Defense at upper left of Figure 3 which also have not yet been treated at this level of detail by the GIG and NCES efforts. However, GIG descriptions of operational capabilities do distinguish audit data, whose major purpose is accountability, and situational awareness data, whose major purpose is to assess the cyber situation in a timely fashion in order to determine an appropriate response.

- Security administration and administrator management have not yet been treated in detail by the GIG/NCES efforts or this architecture study.

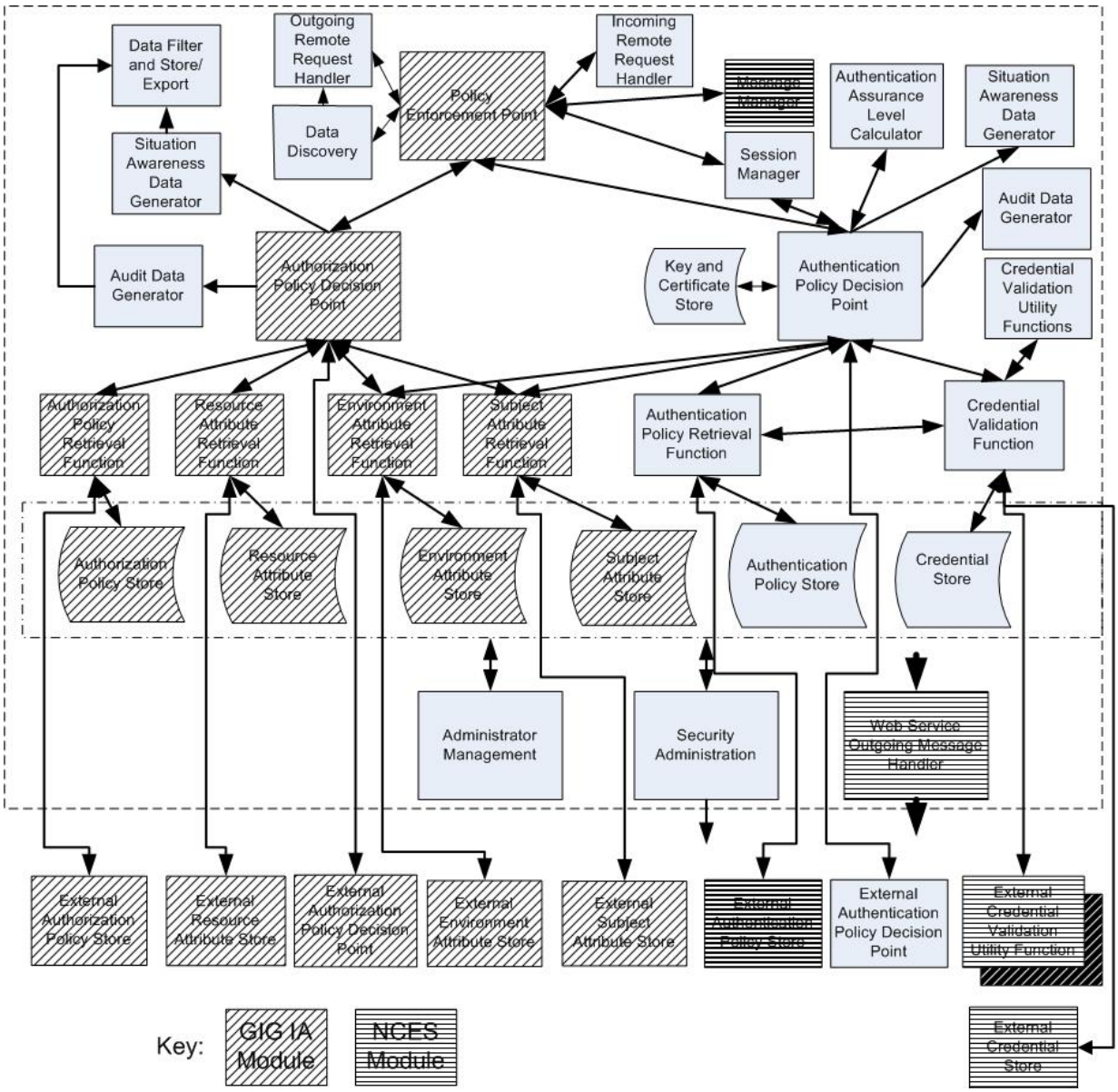

Figure 3 - Leverage of GIG and NCES Architectures

8.0 Issues for Multi-level Security

The need to effectively handle multiple security levels is becoming increasingly important to include in overall security architectures. In the community, the terms multi-level security, multi-security levels, and cross domain are frequently used interchangeably.

For the purpose of our discussions, we will utilize the following definitions[12]:

- Multiple Security Levels (MSL) refers to the use of separate networks to process information from different classification levels or security domains.
- Multiple Level Security (MLS) refers to the use of one network that handles several classification levels or crosses several security domains.
- Cross Domain refers to the exchange of information between classification levels.

The objective of this section is to discuss some of the technology areas and issues as they relate to future MLS environments.

In current multi-level designs, Guard devices are used to separate unclassified and classified networks. There are a number of different types of Guards available but the fundamental operation is similar. The Guards perform functions such as verifying the data sender and recipient and provide content inspection. The process varies depending on whether data is passed from the low to high side or vice versa. Traditionally, content inspection is performed by a human, although some Guards use a more automated means.

The long term goal of the GIG is to provide seamless cross domain information sharing. The data is protected such that only entities that have sufficient privileges will be able to view specific data. This presents many challenges from both usability and protection standpoints. Much of the MLS considerations are handled at the application or operating system level. This requires the use of trusted, MLS operating systems in conjunction with trusted applications. The use of the trusted components influences the security architecture as it represents another level that must be accounted for in the design. In terms of the OIM, this will require some interaction between OS security protections and the ICMS at the application level.

Some of the key issues are described below.

8.1 Data-at-Rest Protection

The use of MLS requires additional protections to be administered more stringently on all assets. This is primarily because the compromise of one asset could have a cascading effect on the

[12] Porell, Jim, Larstan's The Black Book on Corporate Security, www.blackbooksecurity.com.

security of all the data on the machine as well as other machines. It also requires that stored data, or data that is at rest, be appropriately protected.

8.1.1 Data Encryption

Protecting data-at-rest with the use of encryption is another function that is normally performed at the application or operating system level. Most unclassified data-at-rest is not encrypted due to the potential performance sacrifices. One potential method of minimizing the impact on performance is to provide different levels of encryption depending on the level of protection required.

8.1.2 Access Controls

The access control of data is primarily handled at the application layer (e.g., row level security of a database) or by the operating system. The controls must be capable of enforcing various discretionary and mandatory access control approaches. The controls must prevent the ability to modify, copy or delete the data.

8.1.3 Data Labeling and Tagging

The ability to mark or tag data objects for classification level is one of the fundamental issues with MLS. The DoD Meta Data specification along with DCID 6/3 provides guidance as to the appropriate markings. For instance, the DoD Meta Data specification lists the following as an example metadata:[13]

Classification: nonUS
NonUS Classification: NS
Dissemination Controls: EYES
Releasable To: GBR
IC:ownerProducer = ”USA”
IC:classificationReason =”1.4(b)”
IC:classifiedBy =”John Doe, Position Title”
IC:declassDate =”2010-01-01”
IC:SCIcontrols =”ST”
IC:disseminationControls =”REL”
IC:FGIsourceOpen =”AUS NZL NATO”
IC:releasableTo =”USA AUS” />

In reviewing this example data, it is apparent there is no contextual information which makes it extremely difficult to determine the more subjective factors such as the “need-to-know” and limited dissemination (e.g., those which are granted on a case-by-case basis). Some of these subjective factors may be handled by the dissemination controls.

---

[13] DASD, “Department of Defense Discovery Metadata Specification (DDMS), Version 1.2”, 3 January 2005.

The fundamental method of how to apply and protect the data tags is less clearly defined. The current concepts involve binding the metadata to the information such that it can't be separated.

In terms of a security architecture, the cross domain information sharing has several implications. First, the architecture must be extensible in order to handle the additional attributes contained in the metadata objects. Secondly, the decision engine for the access controller needs to be capable of analyzing and approving the requests for data access. This is an extremely complex task as the rationale to approve the requests often depends on the subjective parameter of "need-to-know".

8.1.4 Auditing Capability

A complete auditing capability is necessary to track information objects as they traverse the network. This includes a history of the origin, final destination, and path of the data. This is necessary to identify a chain-of-custody. The auditing capability must also be protected from compromise.

8.1.5 Data Aggregation

The aggregation of data is another consideration. In this case, the combination of two or more pieces of information can be combined to produce or infer information that is at a higher classification. This is extremely difficult to protect against as there are literally unlimited combinations of data. This is further complicated by the fact that data can change levels depending on its context. Inspection is further hampered by the protection mechanisms, such as encryption, which by their nature are to prohibit content monitoring.

8.1.6 Data Integrity

The integrity of the data is another concern, regardless of the classification of the network. The data and metadata need to be protected such that they can't be altered or that any alteration is recorded.

8.1.7 Federation and/or Coalition Partnering

The sharing of information between coalition partners increases the complexity of the MLS issue, and in fact can be considered a sub-set of the MLS problem. There are a number of problems ranging from procedural issues such as how to handle key exchanges to policies problems. The latter is challenging because of the variances in security levels within the Government. For instance, Department of Energy and Department of Defense clearances do not necessarily coincide. Coalition partnering adds another level of complexity in that data may be protected by NOFORN clauses and export control procedures as opposed to classification levels. In some instances, this is almost a more complex issue with respect to dynamic nature of coalitions. Furthermore, certain groups are allowed to share information due to specific export treaties, licenses, or State Department waivers.

8.2 Data-in-Transit Protection

Protecting data in transit has more direct impacts on the security architecture than protecting the data at rest. A few of the concepts are provided below.

8.2.1 Source authentication

Part of the concept of assured information sharing is to perform source authentication. This implies that all connection requests and data sources must be authenticated by the architecture. This requires that protection measures to prevent spoofing, masquerading, and data interception must be implemented.

8.2.2 Connection Security

The ability to protect connections between computers is another important consideration. The currently most popular methods of securing connections is through the use of secure sockets layer (SSL), transport layer security (TLS), or virtual private networks (VPN).

8.2.3 Web Services Data in Transit

In the web services world, such as NCES, the XML_DSIG (XML Digital Signature) and Web Services Security (WSS) are standards that are anticipated to address several of the concerns regarding data protection while in transit. WSS contains a number of standards including:

- WS-Security: SOAP Message Security
- WS-Security: UsernameToken Profile
- WS-Security: X.509 Certificate Token Profile
- WS-SecureConversation
- WS-SecurityPolicy
- WS-Trust
- WS-Federation
- WS-Federation Active Requestor Profile
- WS-Federation Passive Requestor Profile
- WS-Security: Kerberos Binding
- WS-ReliableMessaging

Each of these standards includes recommended data structures and usages. The definitions are available through www.oasis.org and www.ibm.com. Other OIM research efforts are focusing on the interoperability with the web-services specifications, so our effort will not go into further detail on this subject.

8.3 Current/Emerging MLS Tools and Technologies

There are a number of MLS/MSL components and related technologies available including Guards, databases, and operating systems. There are a number of technologies that are rapidly emerging as commercial vendors are seeking solutions to data confidentiality and sharing restrictions.

The following represents a cross section of the technologies available.

8.3.1 ISSE Guard

AFRL and other organizations have been supporting Guard technology such as ISSE for over a decade. The ISSE Guard functions as a boundary control device that provides the ability to pass data between security boundaries. In general, the Guards function as follows:

- The sender is authenticated using 2-factor authentication (username and password).
- The sending host is authenticated using an LDAP server, known only to the ISSE Guard.
- The sending host (via a client) digitally signs the information and encrypts the data and sends it to the ISSE Guard.
- The ISSE Guard receives the information, decrypts the data, verifies the sender (via the digital signature), inspects the data, searches for dirty words, and performs virus detection.
- If the sender and data are valid, the ISSE Guard digitally signs the data, re-encrypts it, and sends it to the receiving host.

Traditionally, the Guards perform key word searches of the data and much of the inspection from the high-to-low side was performed by manual inspection. The use of XML accelerators is permitting more automated inspection methods to be employed.

8.3.2 IBM Z-Series[14]

IBM produces a series of technologies referred to as the Z-Series that performs MLS tasks. These include items such as a virtual machine and database. Most are NSA accredited MLS databases. The virtual machine provides the ability to isolate users and applications on a machine which supports MSL functionality. The database provides secure, row level access control on the data which enables permissions to be specified for individual pieces of data.

[14] Altmark, Alan, "z/VM Security and Integrity", IBM Corporation, April 2005.

8.3.3 Trusted Oracle[15]

Other trusted database applications, such as the high assurance database from Trusted Computer Solutions, have been built around Oracle. Oracle's label security allows controls to be placed at the row level in a database. This allows a single database to store MSL data. This functionality combined with secure operating systems and web servers provides a secure MSL solution.

8.3.4 Digital Rights Management

There have been rapid developments in Digital Rights Management (DRM) area over the past few years. This area will continue to see rapid changes over the next few years. This applies controls at the application and data level to restrict access and usage. This has become increasingly important in both the Government and commercial sectors. The commercial sector is interested in this from the perspective of protecting copyrighted materials such as music and movies. The Government is using this to restrict access and tracking changes in documents (e.g., Microsoft Office document tracking feature).

8.3.5 Transaction Based Security

Emerging programs such as the Lockheed Martin's World Infrastructure Security Environment (WISE) and Defense Research and Development of Canada (DRDC) SAMPOC program use concepts of transaction based security. This provides control on individual actions (e.g., document access and control) on a case-by-case basis.

8.3.6 Trusted Operating Systems[16]

There are several trusted operating systems available from Sun, Linux (a commercial variant), IBM, and a future Windows version. These operating systems provide a significantly enhanced security features such as policy enforcement and data labeling. They also tend to limit the applications to minimum resources as another means of protection.

8.3.7 Security Enhanced (SE) Linux[17]

The NSA is developing an enhanced version of the Linux kernel referred to as SELinux. The purpose of this effort is to incorporate mandatory access control into the Linux kernel to increase the security provisions. This effort built upon previous work from the NSA, Secure Computing

---

[15] Trusted computer Solutions Provides Customers with Military-Grade Cyber Security While Reducing Hardware, Software and Administrative Costs with Oracle label Security", http://www.oracle.com/customers/profiles/PROFILE2945.HTML

[16] "Trusted Linux", http://www.linux.com/article.pl?sid=04/09/30/210215

[17] http://www.nsa.gov/selinux/info/faq.cfm

Corporation, and the University of Utah. The latter provided enhancements to dynamic security policy enforcement.

SELinux provides support for mandatory access control which is used to confine user programs to the minimum resources necessary to operate. This effectively partitions the OS and does not require a root user. This allows a single system to be used by users with differing security authorizations and controlling access to information with differing security requirements. The key aspects of the system include:

- Controls over Use of "Capabilities"
- Clean Separation of Policy from Enforcement
- Well-Defined Policy Interfaces
- Independent of Specific Policies and Policy Languages
- Independent of Specific Security Label Formats and Contents
- Individual Labels and Controls for Kernel Objects and Services
- Caching of Access Decisions for Efficiency
- Support for Policy Changes
- Controls over Process Initialization and Inheritance and Program Execution
- Controls over File Systems, Directories, Files, and Open File Descriptions
- Controls over Sockets, Messages, and Network Interfaces

It should be noted that this effort is an R&D activity, and SELinux is not considered a "trusted operating system."

8.3.8 Multiple Independent Levels of Security (MILS)[18]

MILS is a collaborative effort between Government, industry, and academia to develop a secure architecture and is considered an essential component of the GIG. MILS combines a separation policy that prevents applications at different security levels from communicating. Furthermore, the kernel-level security functions have been separated into modules that reside outside the kernel. The MILS architecture contains a policy engine referred to as the message routing components. This controls the communications between partitions in the operating system resource space. Guards are used to monitor message contents for each application-level protocol. The modular design of the architecture allows individual components to be analyzed for certification purposes, which makes the entire system more realizable.

[18] http://www.stsc.hill.af.mil/crosstalk/2005/10/0510Harrisonetal.html

8.3.9 Intelligent Routing

The use of intelligent routers can reduce some of the issues associated with MLS in that "protected paths" can be utilized. These routers can be programmed to control the communication paths of sensitive information.

9.0 Process Flows

The goal of the process flows in this appendix is to show the proposed OIM authentication and authorization architecture in operation. Based on an analysis of requirements for this architecture, we have selected key processes for detailed development here.

Process flow information includes:

- Process flow step descriptions
- Process flow diagrams
- Related issues discovered during the analysis
- Summary of key decisions made while designing the process flow, and insights during its development. Insights include highlights of similarities and differences between related scenarios supported by this architecture.

9.1.1 Rationale for Process Selection

This discussion is structured into subsections for authentication, authorization and federation, with several process flows grouped under each section. The federation section covers both authentication and authorization in a federated environment.

The list of processes covered and the rationale for selecting these are:

- Authentication
    - *CAC Authentication for Users* – CAC (Common Access Card) is a standard authenticator for the GIG environment
    - *Authentication for Automated Processes* – Automated processes as publishers and subscribers may be a very large constituency for OIM, goal is to understand the implications and compare to human user case
    - *NCES Thick Client Authentication* – Goal is to understand the impact on OIM of providing a web services interface to clients, which was identified as an architecture driver
    - *Password Authentication* – Goal is to understand how the simplest authentication mechanism is realized under the architecture
- Authorization
    - *Authorize Client* – Basic process
    - *Authorize Client in Web Service Mode* - Goal is to understand the impact on OIM of providing a web services interface to clients, which was identified as an architecture driver

- *Authorize Delivery to Subscriber* – Ensure that architecture treats unique character of subscriptions - that delivery is not triggered by user, but by system

- Federation
  - *Outgoing Remote Request to Federated OIM* – Understand security controls on sending requests to federated partners
  - *Incoming Remote Request to Federated OIM* – Understand security controls on receiving and processing requests from federated partners.

The broadest process area that has not been covered is security administration. This is a critical area. One generally develops operational scenarios first, in order to bring into focus what is to be administered, which we have done in the present document. Then one designs processes for administration, which may in turn suggest improvements to the design of operational processes for ease of administration.

Special cases that the architecture is intended to support have not all been treated, particularly authenticating to OIM via a web services portal (NCES thin client), and handling of federation requests when the target for the request does not accept an asserted authentication. These were felt to be important cases, but more advanced and therefore should be developed in detail after the simpler cases were understood.

9.1.2 Notation and Conventions

Where process flows are nearly identical, we use cross-references to other processes instead of repeating identical information. This allows the reader to easily see the similarities and differences between scenarios.

A word on notation: in the process flow diagrams, each process step is labeled. If a communication between modules consists of a request and immediate response, we have described that interaction as a single step and show one label for both directions of this flow in the process diagram. Otherwise, each direction of data flow is distinguished as a unique labeled step. On a two way data flow arrow with two labels (one in each direction), where possible, we have placed each label nearest the arrow that represents the direction referenced by that label.

9.2 Authentication

9.2.1 Scope

The following are the authentication process flows:

- *CAC Authentication for Users* – shows process via which a user logs in to OIM using a common access card.
- *Authentication for Automated Processes* – shows similarities and differences between the login process for users and for automated processes.

- *NCES Thick Client Authentication* – shows how OIM performs authentication as an NCES-compliant web service, when the OIM client accesses OIM directly (not via a web portal). Note this is not a login operation, as authentication is performed for every web service request.
- *Password Authentication* – shows how OIM performs simple password authentication using this architecture, and compares this to the CAC authentication case.

Note that an overview of how one would realize NCES Thin Client Authentication in this architecture is provided in Section 7.1.2.

Special features of authentication covered by these processes are:

- Use of internal or external attribute stores for user and credential information and validation
- Calculation and use of authentication assurance level
- Use of authentication policy
- User may supply certificate reference instead of certificate
- Interface of the authentication process to audit and other computer network defense processes.

9.2.2 Process Analysis

9.2.2.1 CAC Authentication for Users

9.2.2.1.1 Process Steps

Figure 4 and the following step descriptions define user authentication using a common access card. There are no known differences when using a softcopy of a certificate.

1. A user requests login by sending a request to the Policy Enforcement Point (PEP).
2. The PEP provides the Session Manager with information from this request for later reference during the session. This data includes the IP address of the requester, and possibly other data from the requester such as their platform ID. The Session Manager returns a handle to refer to this session later.
3. The PEP asks the Authentication PDP to authenticate this user, providing information from the initial message from the user, that must indicate that this will be certificate-based authentication, and the detailed protocol to be used must be explicit or implicit in this communication.

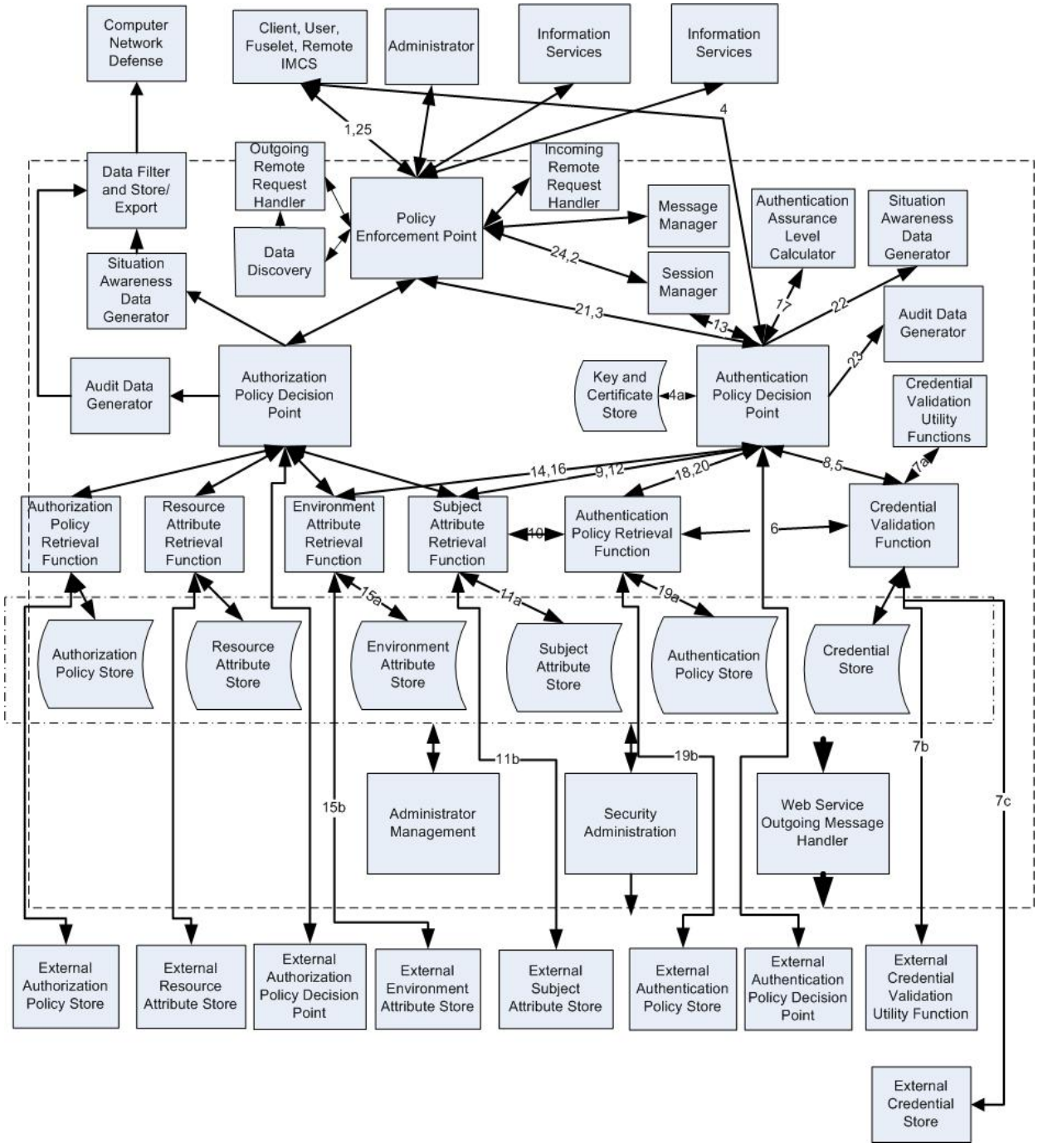

Figure 4 - CAC Authentication for Users

4. The Authentication PDP exchanges a sequence of messages with the client as required by the specific authentication protocol that uses certificates. For example, a user may present their certificate, OIM may validate it and then send the client a challenge (such as a random number), the client may respond by encrypting the challenge with its private key, and the Authentication PDP will then decrypt this with the public key in the

certificate to verify the identity of the user. Use of SSL with the built-in client side certificate based authentication option is also an example, which has considerably more messages in the exchange to create an encrypted session (in addition to providing authentication), but does use the challenge/response idea just described for client authentication. If authentication of OIM to the client using the OIM certificate is part of the protocol exchange, the PDP may consult the Key and Certificate Store to obtain the OIM private key to use during this exchange (4a).

5. At some point before the authentication message exchange with the client completes, the Authentication PDP sends the certificate or certificate reference of the user to the Credential Validation Function. This module determines and coordinates the actions to be taken to validate the certificate.
6. The Credential Validation Function may obtain data from the Authentication Policy Retrieval Function that determines how it should perform its credential validation. Examples are: which certification validation service to use, whether to perform a CRL check.
7. The credential validation function coordinates necessary internal and external utility functions to achieve validation of the certificate. In the short term, actions to verify a certificate will be to invoke an internal certificate path validation function (7a) and call an external service to check the certificate for revocation (7b). If a user provides a certificate reference instead of a certificate, the Credential Validation Function will call out to an external credential store (7c) to obtain the certificate before performing the certificate path validation.
8. The Credential Validation Function reports the results of its check to the Authentication Policy Decision Point.
9. Assuming the certificate verification succeeded, and all other aspects of the authentication protocol message exchange completed successfully, the Authentication Policy Decision Point requests identity status (e.g. active, suspended) and other identity attributes such as strength of mechanism for identity proofing, from the Subject Attribute Retrieval Function.
10. Via the Authentication Policy Retrieval Function, the Subject Attribute Retrieval Function consults the Authentication Policy Store to determine authorized sources for subject information.
11. The Subject Attribute Retrieval Function obtains subject information from local or external sources, or a combination of both.
12. The relevant subject attribute data is returned to the Authentication Policy Decision Point.
13. The Authentication Policy Decision Point requests and obtains data about the session that will be required as an interim step to obtain user environment attributes that impact authentication assurance level. Examples of this type of session data are the platform ID and IP address related to the request.

14. The Authentication Policy Decision Point requests user environment attributes for the authentication assurance level calculation, such as the security status of the IP address of the user's network, and the security posture of the user's platform.
15. The Environment Attribute Retrieval Function determines how to obtain the requested data, and requests and obtains it from local or external sources, or a combination of both.
16. The Environment Attribute Retrieval Function returns the requested data to the Authentication Policy Decision Point.
17. The Authentication Policy Decision Point requests an authentication assurance level calculation, providing certificate, environment and identity-related parameters. The Authentication Assurance Level Calculator does this calculation, and returns the result to the Authentication Policy Decision Point.
18. The Authentication PDP asks the Authentication Policy Retrieval Function for policy information indicating it has a valid authentication, providing certificate information and authentication assurance level as well as the retrieved environment and subject attributes.
19. The Authentication Policy Retrieval Function determines how to retrieve the relevant policy, whether from a local store, a global store, or a combination of both. The relevant policy or policies are returned to the Authentication Policy Retrieval Function.
20. The Authentication Policy Retrieval Function returns the relevant policy information to the Authentication Policy Decision Point.
21. The Authentication Policy Decision Point returns a yes/no answer for the authentication to the Policy Enforcement Point, based on the Authentication Policy information. This implies that the Authentication PDP performs any reconciliation of policy information received from local and external sources. The Authentication Policy Decision Point also returns data from its actions that may be used later in authorization policy, such as authentication assurance level, data from the certificate, session role, and subject and environment attributes.
22. The Authentication Policy Decision Point sends data regarding this authentication, as required, to the Situation Awareness Data Generator, for immediate export and/or analysis.
23. The Authentication Policy Decision Point initiates creation of an audit record for this authentication. (More detail on steps that occur after this step and the previous step can be found in the Authorize Client Action process flow.)
24. The Policy Enforcement Point consults the Session Manager to obtain information about returning a result to the client, such as IP address. The PEP also informs the Session Manager that this client has successfully authenticated, and stores the authentication assurance level, and other data that may be used later for authorization.
25. The Policy Enforcement Point returns the authentication result to the client, along with the handle to identify this logged in session for future interactions

9.2.2.1.2 Flow with Data Elements

In this section we describe the same steps in the above flow, but from the point of view of data elements involved in these flows. Specific data elements are highlighted in bold **courier** typeface.

1. The user will present their **GIG user ID** and possibly **session role** and **authentication-protocol-specific data** elements such as **certificate**. The **authentication protocol type** may be expressed as a data element or may be implicit from other characteristics of this request. The API should support collecting a flexible (by user) and extensible data elements on the front end, such as **IP address** and **platform ID** that may be used to apply authentication or authorization policies. Another future use of the capability could be for obtaining a **quality of service preference** from the user.
2. Minimum data saved for sessions at login request time is **GIG user ID**, **time** and **IP address**. If provided by the client, context data such as **platform ID** and **session role** are saved. A **session ID** is returned.
3. PEP provides **GIG user ID**, **session ID, authentication protocol type,** and any **authentication-protocol-specific data** from the initial user request, for example a **certificate**.
4. The protocol-specific exchange between the Authentication PDP and the client may involve **authentication-protocol-specific data** such as an authentication **challenge** presented by OIM to the client, for the client to return in encrypted form.
5. The **certificate** or **certificate reference** are provided for validation. If the user provided a **certificate reference** instead of a **certificate**, note that NCES has specified a desired format for a certificate reference that should be taken into account. Note we assume that the Credential Validation Function doesn't get any hints from the PDP about how to go about validating this certificate. A different alternative supports a data flow of "hints" from the client, e.g., about what CRL store to go to.
6. This step assumes that Authentication Policy includes data that seems more like configuration – e.g. **External Certificate Validation Utility Function ID,** which is the ID of an authorized service. This is consistent with the GIG concept that identifying external authorities to be trusted with aspects of authentication is part of authentication policy. Here we are suggesting that there are at least two additional parameters of authentication policy. One determines how a certificate is validated (**certificate validation process**), and the other is used later in this process and impacts directly whether the overall authentication returns a positive (**authentication decision rule**). We do not assume that the system will perform the same steps to validate all certificates. The steps will be determined by

an algorithm whose input is the certificate information received, and the authentication policy.

7. OIM may send a **`certificate reference`** to the External Credential Store, which returns a certificate. OIM sends a **`certificate`** to a Certificate Validation Utility function, which returns all or part of the **`certificate validation result`** for this certificate – specifically whether the certificate is expired, or has a valid path, or both.
8. The Credential Validation Function returns a possibly simplified form of **`certificate validation result`**, along with data it has parsed from the certificate that may be used by the process such as **`classification level`** of the user, **`certificate strength`**, etc..
9. The **`GIG user ID`** is sent to the Subject Attribute Retrieval Function.
10. The authentication policy is consulted for the one or more instances of an authorized **`Subject Attribute Store ID`**.
11. For human user subject attributes as defined by the GIG, see Section **Error! Reference source not found.**.
12. These subject attributes are returned to the Authentication PDP.
13. The Authentication PDP presents the **`session ID`** and obtains session data discussed under Step 2.
14. The name of one or more environment entities such as a **`platform ID`** is sent to the attribute retrieval function.
15. Environment data is likely to be in the form of named entities plus a set of attributes. For example, for platform ID, returned attributes might be **`system robustness`** and **`platform location security posture`**.
16. Environment attributes are returned to the Authentication PDP.
17. The **`authentication assurance level`** result, and also the algorithm itself should be represented as data, as it is anticipated to evolve over time. Data that is input to this algorithm is user environment attributes as described in Step 15, **`authenticator type`**, **`identity proofing strength of mechanism`**, other subject attributes such as **`clearance level`**, and for certificates, **`certificate strength`**.
18. The Authentication PDP presents all data about this request gathered to this point, to the Authentication Policy Retrieval Function to obtain a match.
19. The **`Authentication Policy Store ID`** and associated location are required in order to locate the **`authentication decision rule`** for the situation at hand. Note that the architecture assumes that local and global policies include information about how to reconcile them, and that the Authentication PDP does this function.

20. Return of yes/no decision, or applicable **`authentication decision rules`** if reconciliation of policies from various sources is required.
21. The Authentication Policy Decision Point provides the PEP with a yes/no answer for its authentication decision, plus **`authentication assurance level`**, data from the certificate such as **`clearance level`**, and subject and environment attributes as noted in Steps 11 and 15.
22. Data needed for real time or near-real time cyber security situation analysis for OIM or any portion of the GIG, is expected to be more compact and statistics oriented than audit data (which is event oriented). Thus for example, one might update a count of failed authentications per time period. A generic structure for this data is a topic for future study.
23. The audit data for an authentication attempt would include at a minimum an **`event type`**, **`time`**, **`GIG user ID`**.
24. The PEP informs the Session Manager that the login was successful, and provides **`authentication assurance level`**, data from the certificate such as **`clearance level`**, and subject and environment attributes as noted in Steps 11 and 15. Note that based on this data, the Session Manager can recognize multiple logins for one user, an identified requirement on the architecture.
25. The authentication result is a yes or no, plus, if yes, the **`session ID`**.

9.2.2.1.3 Related Issues

- **Assurance for policy decision inputs:** It should be noted that basing policy on data such as platform ID, assumes an underlying infrastructure that makes this information trustworthy as received from the client. We have not designed such an infrastructure in this document.
- **Credential validation flexibility:** Although this decision does not impact the architecture, it remains to be determined what run-time flexibility the Credential Validation function needs in supporting different courses of action for validating the certificate. To be completely general, should one allow its course of action be totally determined by what is in the certificate, plus the authentication policy? For example, could it be the case that some certificates should be verified by the internal function and some by an external function, for some reason? Or, that a different credential store is required to retrieve some certificates than others?
- **RAdAC heuristics and manual review of policy decisions:** The GIG RAdAC concept visualizes that operational need might override other policy parameters based upon heuristic algorithms in the future, but such an override would be subject in some cases to manual review. This OIM architecture does not incorporate such a function. Full implementation of this concept in GIG plans is in 2016.

9.2.2.1.4 Key Decisions and Insights

- **Flexible Authentication Protocol:** The architecture does not specify a specific message exchange for certificate based authentication. It does specify that functionality within the Authentication PDP will be able to recognize and execute each specific protocol to be used. The Authentication PDP is internally comprised of individual components that handle the desired protocols for various authentication methods. This breakdown is not explicitly shown at the level of our diagrams. The Authenticator Design Pattern at http://jerry.cs.uiuc.edu/~plop/plop99/proceedings/Fernandez4/Authenticator3.PDF provides ideas on how to implement this kind of flexibility. We present a concept for this in Section **Error! Reference source not found.**.
- **Order of operations:** We elected the ordering: credential validation, subject attribute retrieval/check, environment attribute retrieval, authentication assurance level calculation, authentication policy retrieval and finally authentication decision. The rationale for this order is as follows:
    - You need all the data from prior operations to calculate the authentication assurance level.
    - There is no point in obtaining the environment attributes if the user has already failed authentication for other reasons.
    - Checking for an active GIG user (subject attribute retrieval) or validating the user certificate seemed to be OK in either order.
- **Smart discovery, dumb retrieval, smart reconciliation:**
    - *Smart discovery.* We elected to place the intelligence to figure out what attribute and policy data to retrieve, and where it is, down in the retrieval functions, instead of having the Authentication PDP figure this out. This makes the Authentication PDP a decision and coordination authority, with intelligence about types of data residing at a lower level. This makes it easier to add and remove retrieval functions from the architecture as self-contained units.
    - *Dumb retrieval.* On the other hand, is it realistic that the retrieval function is smart enough to know what to go get and where, but too dumb to reconcile what comes back from its various resources? We made the retrieval functions do only retrieval, because reconciliation of policies or attributes seemed to belong at a higher level where data other than that just about this particular kind of attribute, could be taken into account.
    - *Smart reconciliation.* Thus the Authentication Policy Decision Point reconciles raw data retrieved by its subservient functions. The algorithm for this should not impact the architecture, as long as the input required to do this reconciliation resides in the inputs to which the Authentication PDP has access.
- **How does smart discovery work?** How would a retrieval function figure out to go out to a specific COI policy service or store? This might not be a matter of great concern for a specific application, but for OIM to operate conveniently as a framework to support

many applications, it becomes a key issue. See Section 6.4.2 for a discussion of this point.

- **Credential Validation direct access to Authentication Policy Retrieval Function:** It did appear unusual to have both the Authentication PDP and the Credential Validation Function both access the Authentication Policy Retrieval Function. Understanding that external CRL resources might not always be available to mobile OIM installations, we allowed for conditional bypass of a CRL check when validating certificates, by postulating an authentication policy addressing this issue, and having the Credential Validation Function access and implement this policy directly. An alternative would be to have Credential Validation return a "no CRL check available" to the Authorization PDP, and then have the Authorization PDP make the decision whether this is OK. We postulated that there may be other policy reasons for certificate validation to proceed in various ways, and that if so, it made sense for the Credential Validation Function (as it applies to certificates) to encapsulate this knowledge, rather than the Authentication PDP.
- **Need for efficient authentication to authorization handoff:** The authentication process is accessing the same external stores as the authorization process, and creating a significant amount of data that will also be used by the Authorization Process. The Session Manager holds this data for later use as appropriate.
- **Authentication Assurance Level Calculator lack of independence:** In the proposed architecture, the Authentication Assurance Level Calculator does not collect its own data inputs actively – they are supplied via the Authentication Policy Decision Point. We considered the alternative in which, for example, the Authentication Assurance Level Calculator itself retrieved environment elements (such as user platform ID) that it needed from the Session Manager, and then went out to the Environment Attribute Retrieval Function to obtain security attributes of these environment elements. The other factors that go into authentication assurance level such as user clearance, other user attributes from the subject attribute store, and strength of credential, it appears might be used independently in a policy aside from their contribution to authentication assurance level. Therefore the Authentication PDP had the need for these and thus could give them to the Authentication Assurance Level Calculator. Even though the design chosen means that new input data to authentication assurance level will require changes to the Authentication PDP, this was preferred to complicating the authentication assurance level calculator and potentially gathering the same data twice, with possibly different results due to timing.

9.2.2.2 Authentication of External Automated Process as Client

9.2.2.2.1 Process Steps

Since there are few differences, we copy here the steps from Section 9.2.2.1.1 for user authentication with a CAC, and annotate the differences for the case in which an automated process is authenticating to OIM.

1. A user requests login by sending a request to the Policy Enforcement Point (PEP). *Change "user" to "process."*

2. The PEP provides the Session Manager with information from this request for later reference during the session. This data includes the IP address of the requester, and possibly other data from the requester such as their platform ID. The Session Manager returns a handle to refer to this session later. *A different set of data may be supplied by automated processes than by users. This should not be a problem if the system can support different sets of data from different human users, which we anticipate will be required due to different client capabilities and needs.*

3. The PEP asks the Authentication PDP to authenticate this user, providing information from the initial message from the user, that must indicate that this will be certificate-based authentication, and the detailed protocol to be used must be explicit or implicit in this communication. *Change user to "process."*

4. The Authentication PDP exchanges a sequence of messages with the client as required by the specific authentication protocol that uses certificates. For example, a user may present their certificate, OIM may validate it and then send the client a challenge (such as a random number), the client may respond by encrypting the challenge with its private key, and the Authentication PDP will then decrypt this with the public key in the certificate to verify the identity of the user. Use of SSL with the built-in client side certificate based authentication option is also an example, which has considerably more messages in the exchange to create an encrypted session (in addition to providing authentication), but does use the challenge/response idea just described for client authentication. If authentication of OIM to the client using the OIM certificate is part of the protocol exchange, the PDP may consult the Key and Certificate Store to obtain the OIM private key to use during this exchange (4a). *No change.*

5. At some point before the authentication message exchange with the client completes, the Authentication PDP sends the certificate or certificate reference of the user to the Credential Validation Function. This module determines and coordinates the actions to be taken to validate the certificate. *Change user to "process."*

6. The Credential Validation Function may obtain data from the Authentication Policy Retrieval Function that determines how it should perform its credential validation. Examples are: which certification validation service to use, whether to perform a CRL check. *No change. Note it will be required (as we see below), that the process be identified as an automated process by an attributes in its certificate. Therefore the Credential Validation Function may use this knowledge to determine how to perform the credential check.*

7. The Credential Validation Function coordinates necessary internal and external utility functions to achieve validation of the certificate. In the short term, actions to verify a certificate will be to invoke an internal certificate path validation function (7a) and call an external service to check the certificate for revocation (7b). If a user provides a certificate reference instead of a certificate, the Credential Validation Function will call

out to an external credential store (7c) to obtain the certificate before performing the certificate path validation. *Change user to "process."*

8. The Credential Validation Function reports the results of its check to the Authentication Policy Decision Point. *No change.*

9. Assuming the certificate verification succeeded, and all other aspects of the authentication protocol message exchange completed successfully, the Authentication Policy Decision Point requests identity status (e.g. active, suspended) and other identity attributes such as strength of mechanism for identity proofing, from the Subject Attribute Retrieval Function. *Although it is theoretically possible, it is unlikely this data flow will exist for some time, for automated processes. A measure of how carefully certificates for automated process are issued, and a measure of how well the managing organization for an automated process protects its private key, is plausible but involves significant new procedures. Even the concept of identity is also not clear for an automated process. Is the next release of the program the same identity – and what if the process changes hands in terms of managing organizations? So either the Authentication Policy Decision Point does not make this request for automated processes, or if it is made, different kinds of data will be available than for human users. Note this implies that automated processes must be distinguishable by the system. The only trusted way to do this is to place this information as an attribute in the certificate. One must make sure that a human user cannot find a way to pretend to be an automated process and thus bypass the subject attribute inquiry.*

10. Via the Authentication Policy Retrieval Function, the Subject Attribute Retrieval Function consults the Authentication Policy Store to determine authorized sources for subject information. *See comments for step #9.*

11. The Subject Attribute Retrieval Function obtains subject information from local or external sources, or a combination of both. *See comments for step #9.*

12. The relevant subject attribute data is returned to the Authentication Policy Decision Point. *See comments for step #9.*

13. The Authentication Policy Decision Point requests and obtains data about the session that will be required as an interim step to obtain user environment attributes that impact authentication assurance level. Examples of this type of session data are the platform ID and IP address related to the request. *No change, though as noted above in step 2, the data involved for automated processes could be different. An OIM application may want to allow all process publishers from specific networks and exclude others, for example. For such a policy, there may be a desire for some projects to implement a local version of environment attributes for automated processes, even if it is not available as a GIG service.*

14. The Authentication Policy Decision Point requests user environment attributes for the authentication assurance level calculation, such as the security status of the IP address of the user's network, and the security posture of the user's platform. *Change "user" to "process."*

15. The Environment Attribute Retrieval Function determines how to obtain the requested data, and requests and obtains it from local or external sources, or a combination of both. *No change.*

16. The Environment Attribute Retrieval Function returns the requested data to the Authentication Policy Decision Point. *No change.*

17. The Authentication Policy Decision Point requests an authentication assurance level calculation, providing certificate, environment and identity-related parameters. The Authentication Assurance Level Calculator does this calculation, and returns the result to the Authentication Policy Decision Point. *If such a calculation is done for automated processes, the calculator will need to be informed that the subject is an automated process, as it is certain that the calculation will be different than for human users. As noted above, such a calculation is theoretically feasible, but not likely in the near term. The underlying concepts are difficult for automated processes - identity, identify proofing strength of mechanism, and measuring how well a managing organization manages a private key used by an automated system.*

18. The Authentication PDP asks the Authentication Policy Retrieval Function for policy information indicating it has a valid authentication, providing certificate information and authentication assurance level as well as the retrieved environment and subject attributes. *No change. However, note that the fact that this is an automated process must be passed as part of the request, since policies written for automated processes will be different than those for human users. We noted above that this piece of information must be part of the process's certificate data.*

19. The Authentication Policy Retrieval Function determines how to retrieve the relevant policy, whether from a local store, a global store, or a combination of both. The relevant policy or policies are returned to the Authentication Policy Retrieval Function. *No change.*

20. The Authentication Policy Retrieval Function returns the relevant policy information to the Authentication Policy Decision Point. *No change.*

21. The Authentication Policy Decision Point returns a yes/no answer for the authentication to the Policy Enforcement Point, based on the Authentication Policy information. This implies that the Authentication PDP performs any reconciliation of policy information received from local and external sources. The Authentication Policy Decision Point also returns data from its actions that may be used later in authorization policy, such as authentication assurance level, data from the certificate, session role, and subject and environment attributes. *No change.*

22. The Authentication Policy Decision Point sends data regarding this authentication, as required, to the Situation Awareness Data Generator, for immediate export and/or analysis. *No change, but the data sent will be different for an automated process than for a human user.*

23. The Authentication Policy Decision Point initiates creation of an audit record for this authentication. (More detail on steps that occur after this step and the previous step can be found in the Authorize Client Action process flow.) *No change, but the data sent will be different for an automated process than for a human user.*

24. The Policy Enforcement Point consults the Session Manager to obtain information about returning a result to the client, such as IP address. The PEP also informs the Session Manager that this client has successfully authenticated, and stores the authentication assurance level, and other data that may be used later for authorization. *No change.*

25. The Policy Enforcement Point returns the authentication result to the client, along with the handle to identify this logged in session for future interactions. *No change.*

9.2.2.2.2 Related Issues

- **Identification and authentication infrastructure for automated processes:** As discussed in Step 9, an identity and credential management infrastructure to support identification and authentication of automated users is a significant challenge, and differs significantly from that for human users.

9.2.2.2.3 Key Decisions and Insights

- **Certificate for automated process needs to identify it as such***:* The certificate is the only trusted source for this information when the system needs it. Without this distinction, authentication and authorization policies intended for human users would also be applied to automated processes, and this is unlikely to make any sense. Also, the fact that we have an automated process is likely to be a parameter that impacts the authentication process flow logic by eliminating some steps. These steps will be implemented for human users long before they are available for automated processes.

- **Subject attributes and authentication assurance level least likely to have clean application for automated processes in the near term:** This is because the identity of a process and measuring how much we really trust its certificate, are new and untested ideas. There may be simpler ways to achieve the desired result.

- **Environment attributes have natural applications for automated processes:** In particular, the location in the network for an automated publisher or subscriber could influence authentication policy (as well as authorization policy) for that process.

- **Process flow for users also supports automated processes:** The difference between certificate authentication for an automated process and a human user is that some steps are likely to be eliminated for the case of an automated process. If all steps remain, they will involve different data and calculations than for human users.

9.2.2.3 Password Authentication

9.2.2.3.1 Process Steps

Figure 5 and the following step descriptions describe user authentication using a password.

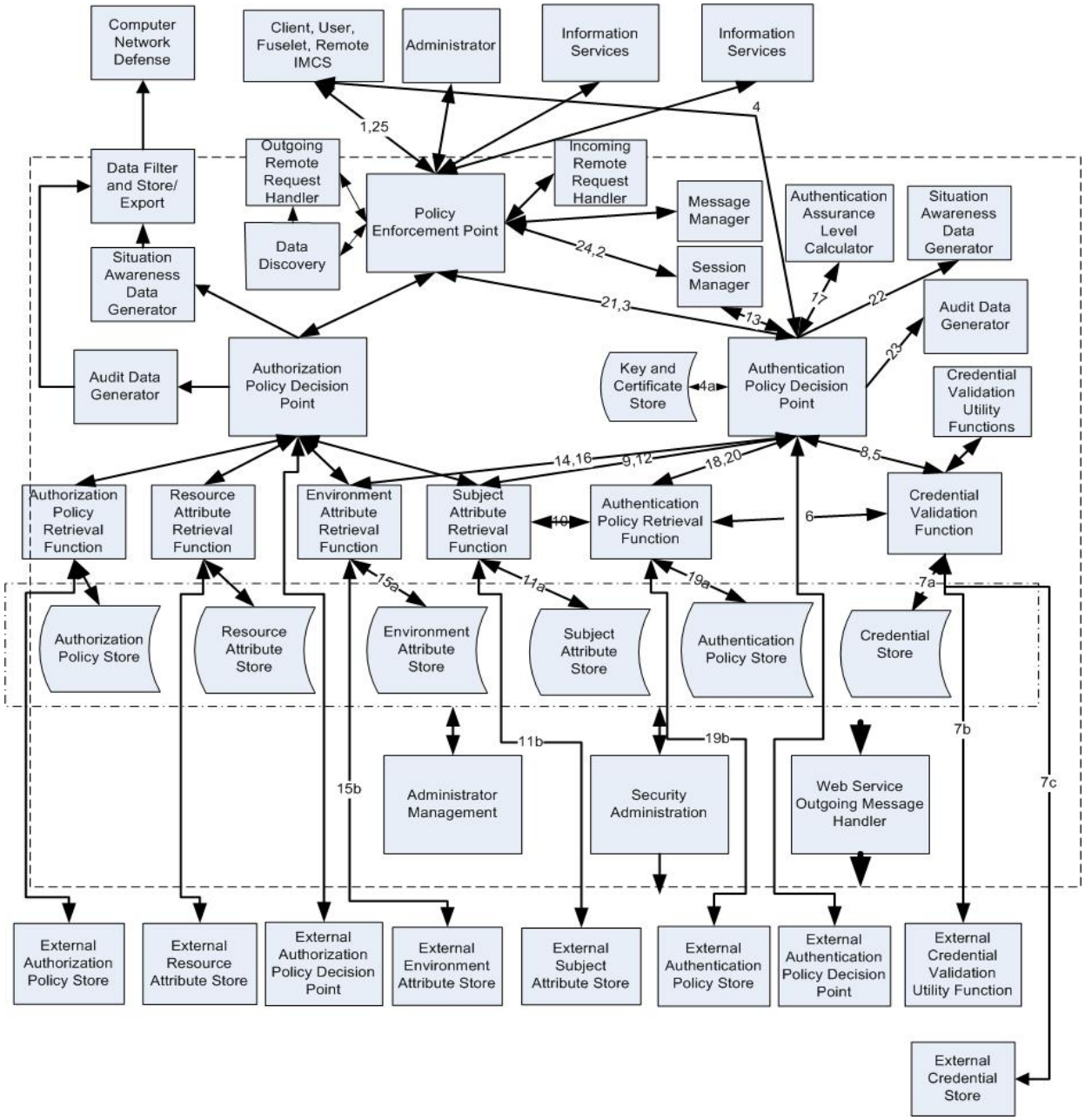

Figure 5 - Password Authentication for Users

1. A user requests login by sending a request to the Policy Enforcement Point (PEP).
2. The PEP provides the Session Manager with information from this request for later reference during the session. This data includes the IP address of the requester, and

possibly other data from the requester such as their platform ID. The Session Manager returns a handle to refer to this session later.

3. The PEP asks the Authentication PDP to authenticate this user, providing information from the initial message from the user, that must indicate that this will be password-based authentication.
4. The Authentication PDP exchanges a sequence of messages with the client to ask for and receive the password.
5. The Authentication PDP sends the password received to the Credential Validation Function. This module determines and coordinates the actions to be taken to validate the password.
6. The Credential Validation Function may obtain data from the Authentication Policy Retrieval Function that determines how it should perform its credential validation. An example is: which Credential Stores containing passwords to reference.
7. The Credential Validation Function determines whether the password is valid by consulting internal (7a) and/or external (7c) Credential stores containing user passwords, or sending the password to an external service requesting a validation (7b).
8. The Credential Validation Function reports the results of its check to the Authentication Policy Decision Point.
9. Assuming the password verification succeeded, the Authentication Policy Decision Point requests identity status (e.g. active, suspended) and other identity attributes such as strength of mechanism for identity proofing, from the Subject Attribute Retrieval Function.

10-25. Same as for CAC Authentication for User, Section 9.2.2.1.1, deleting references to certificate information.

9.2.2.3.2 Related Issues

None identified.

9.2.2.3.3 Key Decisions and Insights

- **Comparison to CAC Authentication for User:** The sequence of steps for password based authentication is nearly identical to that for CAC authentication for user in Section 9.2.2.1. The only differences are that because of the simplicity of the password-based authentication protocol, no validation utilities are needed and the credentials (passwords) themselves are likely to be stored internal to OIM. These differences impact process Step 7. Steps 1-6 and 8,9 shown above are the same as for CAC authentication in principle, and are modified as shown above to clarify their instantiation for password based authentication. Steps 10-25 are identical to the CAC case except that information in a certificate is not available to assist in the authentication decision or in later authorization decisions.

9.2.2.4 NCES Thick Client Authentication

9.2.2.4.1 Process Steps

Figure 6 illustrates the process steps for NCES Thick Client Authentication:

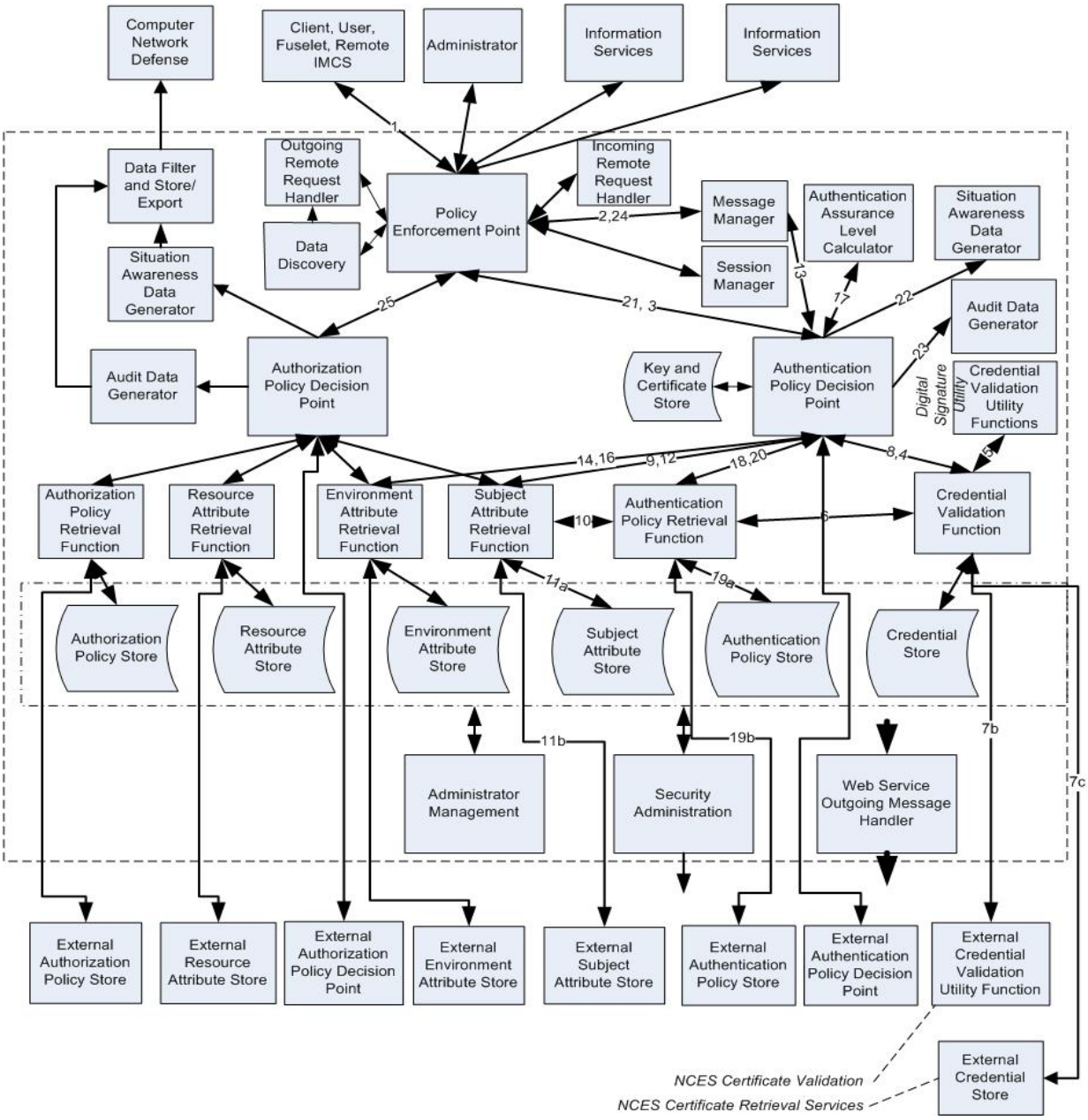

Figure 6 - NCES Thick Client Authentication

The following are process steps for NCES Thick Client authentication, in which OIM acts as a web service to an NCES thick client and OIM uses NCES security services to support

authentication. Note, however, that either of these could be implemented independent of the other.

Prerequisites: Before an OIM client can make a service request of OIM, the OIM instance has (1) obtained a public/private key pair and certificate to be used to identify itself (2) obtained a certificate that contains the public key of the NCES certificate Validation service, that will be implicitly trusted. This NCES service is an instance of the generic external module named "External Credential Validation Utility" in Figure 6.

1. This case is a digital signature check function.
2. The Credential Check Function may obtain data from the Authentication Policy Retrieval Function that determines how it should perform its validation of the certificate. Examples are: which certification validation service to use, and whether to perform a CRL check.
3. In the target NCES environment term, OIM will call the NCES certificate validation service (7b) which will verify the certificate path and check for certificate revocation. If A client sends a request for an OIM service (e.g., to publish, to subscribe, to query) to the Policy Enforcement Point (PEP). The client or user request includes a certificate or certificate-reference to identify the user. Per the NCES architecture, the request is a SOAP message signed with the private key of the client.
4. The PEP provides the Message Manager with information from this request for reference during processing of this message. This data includes the IP address of the requester, and possibly other data from the requester such as their platform ID.
5. The PEP asks the Authentication PDP to authenticate this user, and provides the client's SOAP message as the credential.
6. The PDP sends the provided information to the Credential Validation Function. This module determines and coordinates the actions to be taken to validate the identity of the client using the SOAP message.
7. The Credential Validation Function requests a digital signature check on the SOAP message, and receives a response from a Credential Validation Utility Function, which in a the user provides a certificate reference instead of a certificate, the Credential Validation function should also have the capability to call out to an external credential store or service such as the NCES Certificate Retrieval Service (7c) to obtain the certificate before requesting certificate validation. Note that in order to send this request, the Credential Validation Function will send the request through an Web Service Outgoing Message Handler that will package the request as a SOAP message including the OIM certificate and signed with its private key. Further, the response received from the NCES Certificate Validation service, as with all incoming messages, comes through the OIM PEP, and is subject to the authentication process flow presently being described, as for any other NCES thick client (though for simplicity this is not explicitly shown in the flow diagram), with one difference. It is clear that one cannot ask the NCES Certificate Validation Service to validate its own certificate. Therefore step 7 (this step)

of the authentication process is not performed for messages arriving from the NCES Certificate Validation Service. The initial solution to this issue discussed in the NCES architecture document is for clients of NCES to explicitly trust a pre-distributed certificate for the NCES Certificate Validation Service.

8. See step 8 CAC authentication for users flow in Section 9.2.2.1.1.
9. See step 9 CAC authentication for users flow in Section 9.2.2.1.1.
10. See step 10 CAC authentication for users flow in Section 9.2.2.1.1.
11. See step 11 CAC authentication for users flow in Section 9.2.2.1.1.
12. See step 12 CAC authentication for users flow in Section 9.2.2.1.1.
13. The Authentication Policy Decision Point requests and obtains data about the service request that will be required as an interim step to obtain user environment attributes that impact authentication assurance level. Examples of this type of session data are the platform ID and IP address related to the request. This is the same as step 13 CAC authentication for users flow in Section 9.2.2.1.1, except "session" is replaced by "service request."
14. See step 14 CAC authentication for users flow in Section 9.2.2.1.1.
15. See step 15 CAC authentication for users flow in Section 9.2.2.1.1.
16. See step 16 CAC authentication for users flow in Section 9.2.2.1.1.
17. See step 17 CAC authentication for users flow in Section 9.2.2.1.1.
18. See step 18 CAC authentication for users flow in Section 9.2.2.1.1.
19. See step 19 CAC authentication for users flow in Section 9.2.2.1.1.
20. See step 20 CAC authentication for users flow in Section 9.2.2.1.1.
21. See step 21 CAC authentication for users flow in Section 9.2.2.1.1.
22. See step 22 CAC authentication for users flow in Section 9.2.2.1.1.
23. See step 23 CAC authentication for users flow in Section 9.2.2.1.1.
24. The PEP provides any required update to the Message Manager (it is not clear what is needed at this point).
25. The PEP sends the Authorization Policy Decision Point (PDP) information related to the request, including the ID of the requester, attributes from the certificate, the data object (and its intended metadata if this is a publish operation), the calculated authentication assurance level, and subject attribute and environment attribute data collected during authorization.

It should be pointed out that when acting as a web service for clients, completion of the authentication process directly triggers the authorization process described in Section 9.3.2.2. This is different than the non-web services case, within which authentication is a unique login request from the client to be responded to. In that case, authorization is triggered when the client requests a particular (non-login) action.

9.2.2.4.2 Related Issues

- **NCES Security Services act like thick clients:** It should be noted that this process applies to NCES Security Services themselves, when interacting with OIM. Thus the OIM authorization policy would need to recognize and allow responses from NCES.

9.2.2.4.3 Key Decisions and Insights

- **Overview of architecture impact of NCES Support:** The key differences between the authentication architectures where OIM acts as a web service to clients, and its non-web services interface, are as follows:
    - **Message Manager replaces Session Manager:** Under a web services architecture, there is no concept of login to OIM. Thus there is no concept of a logged in session at all. In the overall flow, a Message Manager replaces the Session Manager to track the state of interactions with clients while processing a request for a service.
    - **SOAP message as credential:** The architecture views the entire SOAP message as a credential to be checked. From the point of view of the Authentication PDP, it is simply passing a particular type of credential to the Credential Validation Function, which figures out how to validate it.
    - **Keys and Certificates:** To support NCES thick or thin clients, or even simply to interact with any NCES services, the OIM architecture requires that an OIM instance have an a public/private key pair and certificate, for signing outgoing SOAP messages. These are in the Key and Certificate Store. To use the NCES Certificate Validation Services, OIM must in addition have a pre-distributed NCES Certificate Validation Service certificate that is implicitly trusted.
    - **Outgoing message packaging:** Whether to support NCES thick or thin clients, or simply to interact with any NCES services, a Web Service Outgoing Message Handler module is required to package up outgoing messages in the required SOAP format, including the OIM certificate or a reference, and to sign the message with the OIM private key.
    - **Action upon completion of authentication:** In the case of CAC authentication for users, the end of the authentication process is the return of an authentication result to the client. Under the web services architecture, authentication is part of an overall message processing flow which immediately triggers the authorization flow, as long as the authentication is successful.

- **Caching for policy driven authentication and authorization:** Despite the above architecture differences, once an incoming signed SOAP message has successfully passed its NCES required checks, the Authentication PDP is free to apply the same additional criteria to pass/fail this authentication as for the CAC authentication case: particularly considering attributes of the subject and environment, and authentication assurance level. Authentication assurance level can also be passed on to the authorization process as for CAC authentication. This transfer occurs directly via the Authentication PDP, instead of via storage in the Session Manager in the case where a login occurs. However, consulting external stores for required attributes and recalculating authentication assurance level for every message may be quite inefficient. A caching strategy will be required for some applications.

- **Caller for digital signature check:** We could have had the Authentication PDP call the digital signature check Credential Validation Utility Function to verify the digital signature on the SOAP message, and then call the Credential Validation Function just to verify the certificate. We did not elect this option since it would build validation specifics into the Authentication PDP that could be handed off to a lower level, keeping the sequence of functions performed by the Authentication PDP generic. Thus the Credential Validation Function calls the digital signature check utility.

- **When to generate situation awareness and audit data:** Earlier versions of the CAC authentication for users flow showed situation awareness and audit data generation occurring after login results were sent to the client. After developing the present flow for the web service case, we modified this. To minimize differences in the sequence of operations for web services and non-web-services cases, we moved generation of this data right after the Authentication PDP sends its response back to the PEP, since this gives us a consistent point in both flows.

Reference: NCES CES (2004). *Net-Centric Enterprise Services (NCES) Security Core Enterprise Services (CES) Architecture* (Version 0.5), 29 December.

9.3 Authorization

9.3.1 Scope

The following are the authorization process flows:

- *Authorize Client Action*– shows the generic process used to check that a client is authorized to perform a specific action on a specific OIM resource

- *Authorize Client in Web Service Mode* – shows similarities and differences between the generic process for authorizing a client, and the process used when OIM is operating as web service

- *Authorize Delivery to Subscriber* – shows the process via which authorization is rechecked before information objects are delivered to a client who has set up a

subscription. We also include general remarks on how changes to client privileges are handled in the architecture.

A client could be a human user or an automated program.

Special features of authorization covered by these processes are:

- Use of internal and external authorization policy
- Use of authentication assurance level, and subject, environment, and resource attributes in determining authorization
- Interface of the authorization process to audit and other computer network defense processes.

9.3.2 Process Analysis

9.3.2.1 Authorize Client Action

9.3.2.1.1 Process Steps

Figure 7 and the following step descriptions describe the Authorize Client Action process.

1. A client requests an action on data object(s) (subscribe, publish, query, submit federated request).
2. The Policy Enforcement Point (PEP) queries the Session Manager to determine if the client is authenticated. This might be implemented, for example, by sending the client a random number on authentication that it returns to the PEP along with subsequent requests. If the user is authenticated, the process continues with the next step. If not, an error message will be returned to the client.
3. The PEP sends the Authorization Policy Decision Point (PDP) information related to the request, including the ID of the requester, attributes from their certificate if certificate authentication was used, the action, the data object (and its intended metadata if this is a publish operation), the calculated authentication assurance level, and subject attribute and environment attribute data collected during authentication. Note that a subset, and perhaps none, of this subject and environment data collected earlier will be useful for authorization, since it may be old information.
4. If the data object that is the target of this action already exists (this is not a publish operation), the Authorization PDP asks the Resource Attribute Retrieval Function to obtain the metadata and any other attributes associated with the data object.
5. The Resource Attribute Retrieval Function determines how to obtain metadata and other attributes for the data object from the internal and external resource attributes stores via 5a and 5b.
6. The Resource Attribute Retrieval Function returns the attributes to the Authorization PDP.

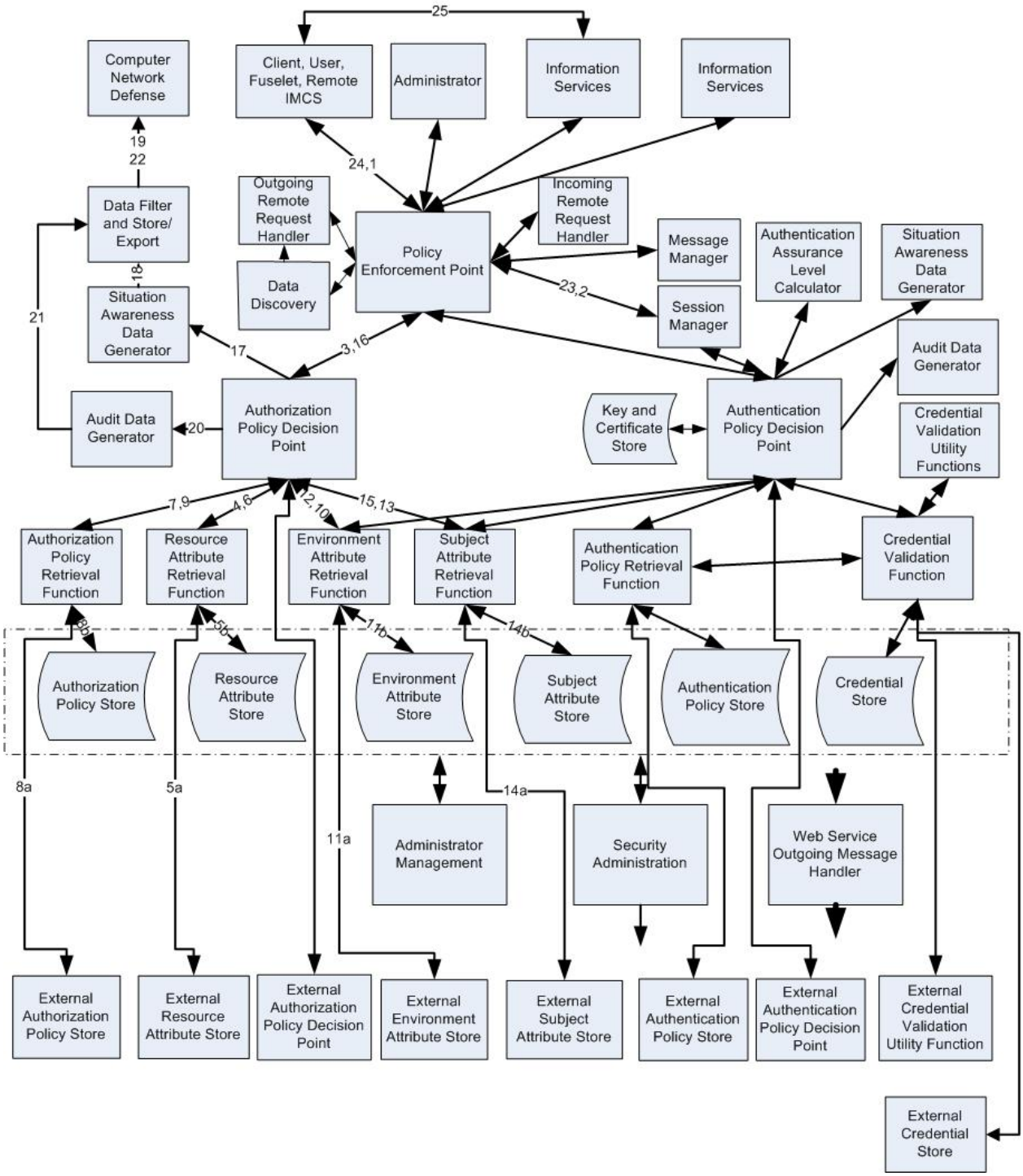

Figure 7 - Authorize Client Action

7. The Authorization PDP requests the Authorization Policy Retrieval Function to obtain all policies that match the parameters collected about this request up to this point.

8. The Authorization Policy Retrieval Function using 8b, consults the Authorization Policy Store to determine whether environment and subject attribute data collected during authentication can be used for this authorization decision, dependent on its data type, and the time since retrieval. The Authorization Policy Retrieval Function then determines how to obtain all policies that might apply to a request with the parameters specified (not including the old attribute data just mentioned if policy dictates) using internal interface 8a and external interface 8b. Note that a single policy match cannot yet necessarily be retrieved, since the set of all attributes that might be relevant to the policy is not known up front by the Authorization PDP.
9. After receiving candidate policies from the Authorization Policy Retrieval Function, the Authorization PDP examines each candidate policy. It determines if it needs to request additional attribute values as called out in the candidate policies, to determine a match.
10. The Authorization PDP requests from the Environment Attribute Retrieval Function, any environment attributes called out in a candidate policy, that it does not yet have. It may have to perform this step several times if no heuristic optimization is applied to obtain all relevant data the first time.
11. The Environment Attribute Retrieval Function determines how to obtain any additional requested environment attributes from internal and external stores via 11a and 11b.
12. The Environment Attribute Retrieval Function returns the requested attributes to the Authorization PDP.
13. The Authorization PDP requests from the Subject Attribute Retrieval Function, any subject attributes called out in a candidate policy, that it does not yet have. It may have to perform this step several times if no heuristic optimization is applied to obtain all relevant data the first time.
14. The Subject Attribute Retrieval Function determines how to obtain any additional requested subject attributes from internal and external stores via 14a and 14b.
15. The Subject Attribute Retrieval Function returns the requested attributes to the Authorization PDP.
16. If the Authorization PDP finds a matching policy, it returns an access permitted message to the Policy Enforcement Point. If it does not, it returns an access denied message to the Policy Enforcement Point.
17. The Authorization PDP informs the Situation Awareness Data Generator that data about this authorization is available.
18. The Situation Awareness Data Generator gathers the situation awareness data and provides it to the Data Filter and Store/Export Function.
19. The Data Filter and Store/Export function determines the required recipients, and applicable data subset and formats for each recipient, for the new situation awareness data. It exports and/or stores these various streams of data to these various destinations.

20. The Authorization PDP informs the Audit Data Generator that data about this authorization is available.
21. The Audit Data Generator gathers the audit awareness data and provides it to the Data Filter and Store/Export Function.
22. The Data Filter and Store/Export function determines the required recipients, and applicable data subset and formats for each recipient, for the new audit data. It exports and/or stores these various streams of data to these various destinations.
23. The Policy Enforcement Point obtains any required information from the Session Manager to be able to return a response to the client.
24. The Policy Enforcement Point informs the user if permission is denied. If permission is granted, the Policy Enforcement Point provides the client a mechanism to perform the requested action on the information object.
25. The Client performs the requested action on the information object via Information Services.

9.3.2.1.2 Related Issues

- **Event history:** KAoS supports a concept of event history, where a history of events may be retained and considered when evaluating authorization policy. We have not explicitly included this concept in this architecture. However, it could be supported by adding an event history store and retrieval function, parallel to the other store/retrieval module pairs included in the architecture.

9.3.2.1.3 Key Decisions and Insights

- **Impact of flexible policies on architecture:** A consequence of a flexible and extensible policy is that other than by examining all policies, there is no way to know what attribute values are needed to find a matching policy. Thus it is not possible to simply collect a fixed set of attributes and do a search for a matching policy, because you don't know what the needed attributes are, and the set is not fixed. There are a number of possible ways to solve this:
    - Examine all the policies first, determine all attributes that could possibly be needed, and collect the values for them all. The policy examination part of this might seem heavy, but might be optimized and cached – however, collecting attributes that are not going to be needed except for very special cases doesn't appear to be an efficient solution, particularly when some of these attributes will require use of external interfaces to retrieve them.
    - Walk through all possible policies, examining each one, collecting attributes as needed for each one, until you find a matching policy. This would mean that the entire contents of the policy stores would need to be stored and searched through by the PDP, if the policy match happened to be last in the list. This is not a scalable solution. It degrades with large numbers of policies. These must not only

be transferred all at once, or in a sequence of interactions between the PDP and internal and external stores, but also must all be examined until a match is found by the PDP.

- Have the PDP obtain most likely relevant policies based on key attributes, then walk through this reduced set to obtain any remaining attribute values it needs to determine a match. In many cases, the request for relevant policies will actually return a policy match that depends only on the attributes collected by authentication, plus the object metadata. More complicated policies are supported but do not slow down evaluation of simpler ones. This is the recommended solution.

- **Order of operations:** We use the GIG assumption that for data objects, metadata will determine authorization policy. To narrow down the candidate policies that the Authorization PDP has to try, metadata is retrieved first, and used as a parameter when returning relevant policies.
- **Attributes collected at authentication get stale:** Even though the authentication process collects data that may be directly referenced by authorization policies (such as nationality or clearance from the Subject Attribute Store), it became clear that even though one might desire to reuse this data for authorization, it may not be always be wise to do so. For example, an individual could have logged in yesterday, and had their clearance increased today. Therefore even though it is desirable to be as efficient as possible in the number of accesses to external stores, one may not be able to completely leverage what happened during authentication at authorization time.

9.3.2.2 Authorize Client Action in Web Service Mode

9.3.2.2.1 Process Steps

Figure 8 and the following steps describe the process flow "Authorize Client Action in Web Service Mode." As discussed below, it is nearly identical to that for Authorize Client Action, except at its beginning, and its end.

1. The PEP sends an authorization request to the Authorization PDP. The first step of this process flow is the same as Step 25 of the process flow NCES Thick Client Authentication (9.2.2.4.1). This is also the same as Step 3, in the process flow Authorize Client Action (9.3.2.2.1). The remark in Step 3, regarding possibly not being able to reuse attribute information collected during the authentication phase for authorization, remains technically accurate but most likely irrelevant, since when the system is acting as a web service, authentication immediately precedes authorization.
2. See step 4 Authorize client action flow in Section 9.3.2.1.1.
3. See step 5 Authorize client action flow in Section 9.3.2.1.1.
4. See step 6 Authorize client action flow in Section 9.3.2.1.1.
5. See step 7 Authorize client action flow in Section 9.3.2.1.1.

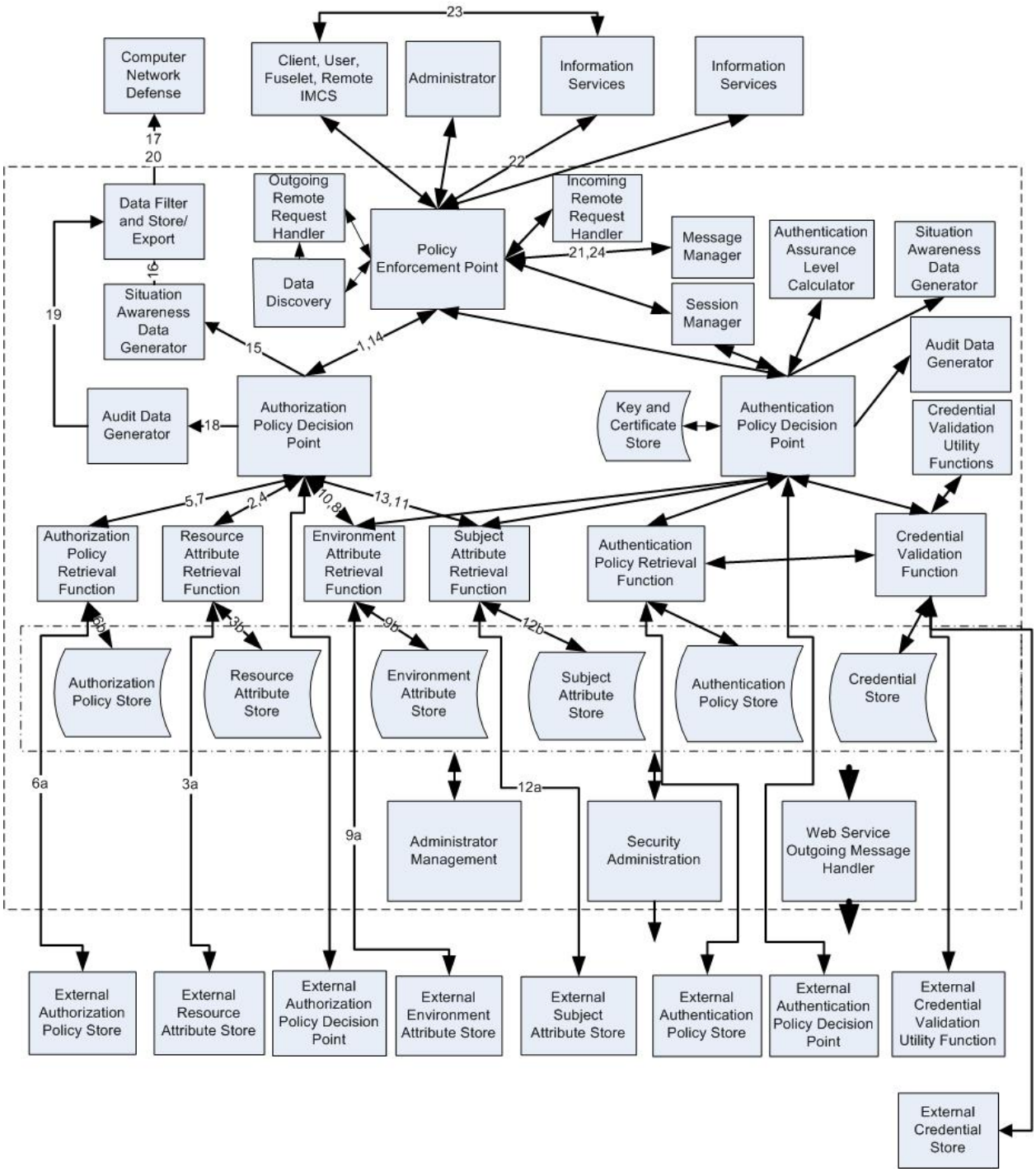

Figure 8 - Authorize Client Action in Web Service Mode

6. See step 8 Authorize client action flow in Section 9.3.2.1.1. Note that the check to see whether any attributes collected during authorization are stale and need to be retrieved again, would logically work, and return a "no," since authentication just occurred.

7. See step 9 Authorize client action flow in Section 9.3.2.1.1.
8. See step 10 Authorize client action flow in Section 9.3.2.1.1.
9. See step 11 Authorize client action flow in Section 9.3.2.1.1.
10. See step 12 Authorize client action flow in Section 9.3.2.1.1.
11. See step 13 Authorize client action flow in Section 9.3.2.1.1.
12. See step 14 Authorize client action flow in Section 9.3.2.1.1.
13. See step 15 Authorize client action flow in Section 9.3.2.1.1.
14. See step 16 Authorize client action flow in Section 9.3.2.1.1.
15. See step 17 Authorize client action flow in Section 9.3.2.1.1.
16. See step 18 Authorize client action flow in Section 9.3.2.1.1.
17. See step 19 Authorize client action flow in Section 9.3.2.1.1.
18. See step 20 Authorize client action flow in Section 9.3.2.1.1.
19. See step 21 Authorize client action flow in Section 9.3.2.1.1.
20. See step 22 Authorize client action flow in Section 9.3.2.1.1.
21. The Policy Enforcement Point obtains any data required to respond to the service request, such as the client IP address, from the Message Manager.
22. Assuming the authorization check has passed, the PEP requests Information Services to respond to the client, providing all required information.
23. Information Services responds to the request, using the

    Web Service Outgoing Message Handler to package its response to the client (though that packaging call is not explicitly shown in the diagram).
24. The PEP cleans up the Message Manager related to this request.

9.3.2.2.2 Related Issues

See Section 9.3.2.1.2.

9.3.2.2.3 Decisions and Insights

- **Overview of architecture impact of NCES Support:** There are few differences between the authorization architecture in which OIM acts as a web service to clients, and its non-web services interface. These are at the beginning and end of the flow:
    - In the web services case, the authorization process is triggered by the completion of the authentication process. This is simply another consequence of the fact that web services operates on service requests, not in terms of logged in sessions.

- Handoff to Information Services after authorization operates differently in the web services and non-web services cases. At completion of authorization, in the web service process flow, the PEP asks Information Services to respond to the client. Upon completion of the authorization process flow in the non-web services case, the PEP provides the client with a handle to allow them to interact with Information Services for the action requested.

- **"Unnecessary" stale attribute check:** For consistency, we retained the logical query of the local policy to see if any of the subject or environment attributes collected during authentication are stale, even though it is not necessary in the web services case since authentication occurs right before authorization. If this check generates any significant overhead, the flow could be modified so that the PEP sends in information indicating that the check can be bypassed, in the web services case.

- **PEP interaction with Message Manager for message response:** An alternative to having PEP interact with Message Manager to determine how to respond to the service request, and then passing this to Information Services, is to have Information Services interact with the Message Manager directly to obtain the required information. With the proposed design, a cleaner assurance argument can be made that Information Services does not contact unauthorized clients, since Information Services has no interface to initiate a request for data about clients to contact. Thus there is less trust in Information Services, and more trust in the PEP, and assurance efforts can be concentrated there. For the same reason, the PEP does the cleanup of the Message Manager once a message has been processed.

9.3.2.3 Authorize Delivery to Subscriber

The actions query or publish are triggered by a client, and per the process flows in the previous two sections, the authorization of the client (including consideration of environment attributes and metadata of the object requested) is assessed in the steps directly before responding to the client (steps 3-16 in Section 9.3.2.1.1). A part of this check, in step 8, is reference to a policy that says whether the authorization process can use attribute data collected during the authentication process, or needs to refresh these. We do not discuss caching of attributes at this level of architecture, however, it is likely that the "freshness" policy referenced in step 8 would be broadened in the case caching is used, to also specify how fresh attribute data needs to be for specific types of transactions.

However, the case of subscription is unique, because the action of delivery to a subscriber is initiated by an earlier subscribe action by a client, but triggered later by the system. This raises a question of how the system insures that the client still has authorization to receive the information from this subscription. That case is covered by this process flow.

9.3.2.3.1 Process Steps

Figure 9 and the following step descriptions describe the Authorize Delivery to Subscriber process.

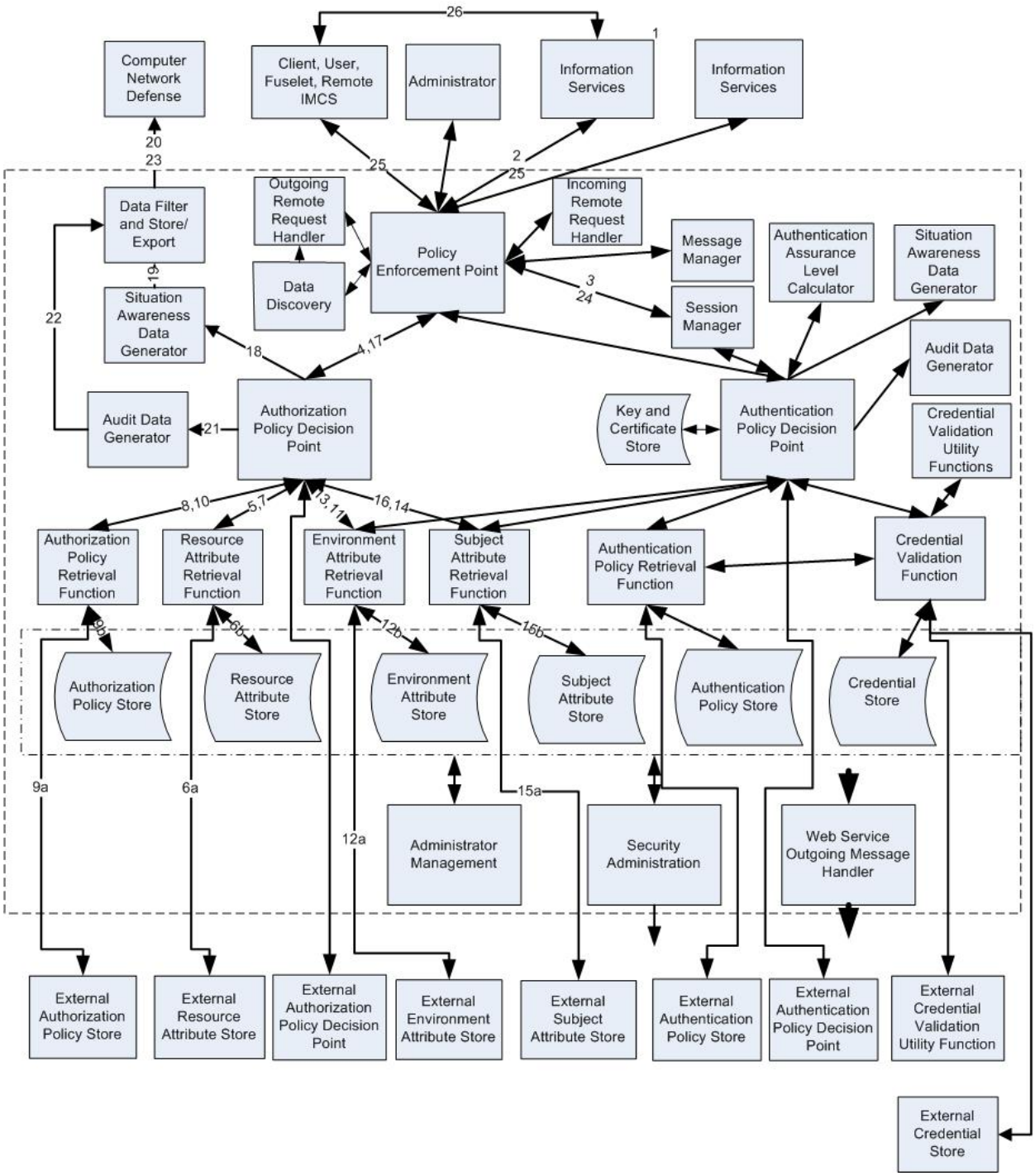

Figure 9 - Authorize Delivery to Subscriber

Prerequisite: A client has an active subscription for information objects satisfying some metadata expression.

1. An action occurs in the system that creates a new object that matches the metadata expression in the client's subscription. This is typically due to a new publication but may be due to a modification of metadata values for an existing information object.
2. OIM evaluates its list of active subscriptions against this new object. If the client's subscription matches the new object, OIM queries the Policy Enforcement Point to determine whether the client still is authorized for a subscription to this particular information object.
3. The Policy Enforcement Point (PEP) queries the Session Manager to determine if the client is logged in.

4-23. The authorization logic proceeds as in steps 3-22 of Authorize Client Action in Section 9.3.2.1.1.

24. The Policy Enforcement Point obtains any required information from the Session Manager to be able to return a response to the client.
25. The Policy Enforcement Point informs the user if permission is denied, and requests OIM to delete the subscription "record" from the system. If permission is granted, the Policy Enforcement Point requests OIM to deliver the information object and gives it the client information to allow it to do so.
26. OIM delivers the information object to the client.

9.3.2.3.2 Related Issues

- **Pre-emptive subscription cancellation:** In this architecture, we do not immediately cancel a subscription if a subscriber loses authorization for this data. This is unnecessary, since authorizations must be checked anyway at the time when information is to be delivered under them. Thus we have effectively delayed cancellation of a subscription until the subscriber is actually about to receive some information under it. This design is secure, but not particularly user friendly. A useful addition would be analysis of the list of active subscriptions against the client authorizations, when changes are made to these authorizations, and the sending of an advisory to the client if authorization is no longer valid, or even automatic deletion of the subscription. Note this would be a relatively complex feature, very useful from a usability standpoint, but system security is not dependent upon its correct operation.

9.3.2.3.3 Key Decisions and Insights

- **Trust and division of functionality – Information Services (IS) module vs. security modules:** In this architecture, the IS module analyzes the list of subscriptions to determine whether a delivery of a specific information object is required. It is responsible for delivery of an information object, identified by its name, to a specified

target, and for maintaining the list of active subscriptions. The security modules are responsible for determining the attributes of an object, the data needed to return it to the client who subscribed to it, and whether this client is still logged in. A first goal in defining this division of functionality is to minimize trust of complicated functions in the IS module. A second goal is to keep the security module functions as small as possible. Under the proposed division of functionality, the IS module is trusted to perform the relatively straightforward action of delivering an information object specified by name to a defined target, when asked to do so by the PEP. Therefore the assurance of code that determines whether the subscriber is still logged in, how to reach them, and evaluates policy, falls under the security modules, since errors in any of these functions could cause an unauthorized delivery. Note that analysis of the list of subscriptions falls under the IS module, since an error in this code could not cause unauthorized information to be delivered under this architecture, though it might cause unwanted information to be delivered, or desired information to not be delivered. Likewise, update of the list of subscriptions can be under the control of the IS module, because even if an unauthorized subscription remains on the list, no associated delivery will occur under this architecture.

- **Information Services to PEP Interface Protection:** The implementation must ensure that only the PEP can request the OIM to deliver information objects to clients. Clearly if this interface was open to all, even perfect assurance of the security modules would not ensure that only authorized information objects are delivered under subscriptions.

9.4 Federation Interactions

9.4.1 Scope

As described in at a high level in Section 6.3.5, the proposed OIM federation architecture can allow a user access to the resources of any OIM instance, from the OIM into which they have logged in. This capability involves the OIM that logged in the user, sending requests to one or more remote OIM's on that user's behalf. The OIM receiving such a request then responds to the requesting OIM, who responds to the user. This operation involves both authentication and authorization for both the sending and receiving OIM instances, as well as the end user. In the case we have detailed here, the remote OIM accepts proof of end user authentication from the requesting OIM. Thus we have the two halves of this operation:

- *Outgoing Remote Request to Federated OIM* – shows how an OIM sends requests to a remote federated OIM on behalf of a client
- *Incoming Remote Request to Federated OIM* – shows how an OIM receiving a request from a remote federated OIM, processes this request.

We have not detailed federation interactions for OIM when it operates as a web service. See Section 6.4.7 for a discussion of this topic.

9.4.2 Process Analysis

9.4.2.1 Process request from federated OIM – accept authentication

9.4.2.1.1 High Level Process Steps

Because the details of this process are lengthy, we begin with a high level overview.

High level process steps:

Prerequisite: Client has been logged in by some OIM PEP.

- **Authorize outgoing request:** If some portion of the data that matches a query or subscription request is not directly accessible through the PEP where the client logged in, the client's OIM sends request(s) out for that data on behalf of the client, to the OIM system(s) where it resides, if allowed by the Authorization PDP associated with the client's PEP.
- **Authenticate system sender and authorize incoming request:** The system that receives such a request authenticates the sending system and accepts the request for further processing, if allowed by the Authorization PDP for the receiving system.
- **Authenticate client or accept authentication:** The authentication policy of the receiving system determines whether the receiving system will accept client authentication information sent by the sending system, or will authenticate the client directly.
- **Authorize client:** The authentication policy of the receiving system also determines whether the receiving system will accept client attribute information sent by the sending system, or will directly obtain these attributes. The receiving system then applies its authorization policy to determine whether the client is authorized for the action requested. If so, it services the client via the requesting OIM.

In summary, this process involves authentication of both the sending system and the client by the receiving system, and authorization at the following three points:

- By the sending system, for the outgoing request to the remote PEP
- By the receiving system, for processing the request received from the sending system
- By the receiving system, for the request by the client.

Steps by an OIM instance to process a client request that requires data accessible via a remote OIM are described in Section 9.4.2.1.2. Steps performed by an OIM receiving such a request, are described in Section 9.4.2.1.3.

9.4.2.1.2 Detailed Process Steps – Outgoing Data Request

Figure 10 and the following step descriptions define the process for an Outgoing Data Request.

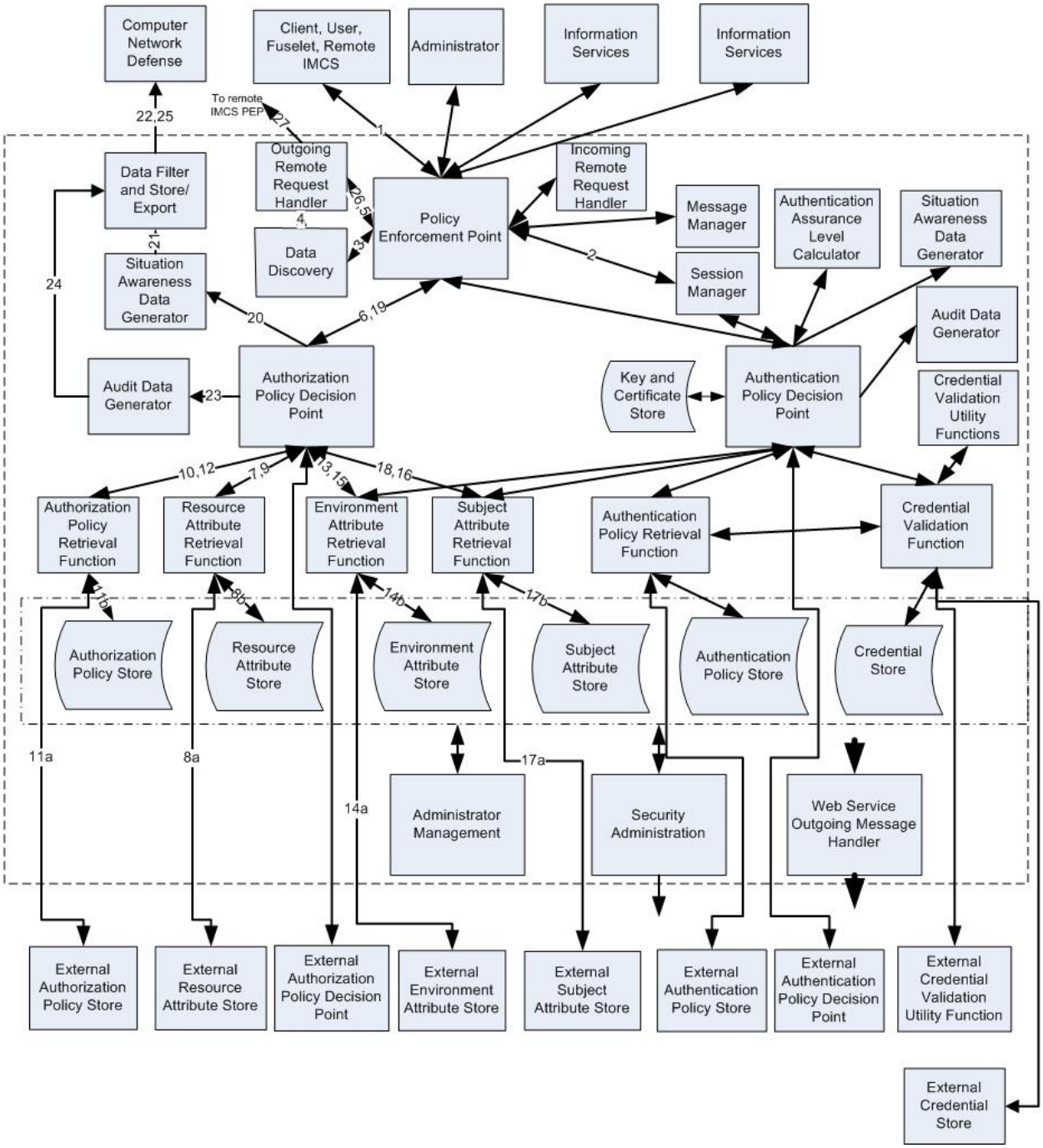

Figure 10 - Outgoing Remote Request to Federated OIM

1. User requests subscription or queries for information via the PEP to which they are logged in. Assume that not all of the information requested is controlled by the PEP to which the user has logged in (the user just makes a request – they are not aware of the location of the data).
2. PEP checks Session Manager to verify that user is logged in.
3. PEP sends the user's request to the Data Discovery module, which determines to which OIM PEP's the request should be routed. It may be routed to more than one, and the set may include the PEP to which the user is logged in, plus other PEP's that control information held in remote Information Services instances. Requests for the local Information Services are routed through the local Authorization PDP and fulfilled as in Figure 7 - Authorize Client Action. This local processing is not shown in the Outgoing Data Request figure below.
4. The Data Discovery module informs the Outgoing Remote Request Handler of the data location(s) and type of request required.
5. The Outgoing Remote Request Handler informs the PEP of each external request that it intends to send to fulfill the user's request.
6. For each of these requests, the PEP queries the Authorization PDP to determine whether it is authorized.

7-19. The Authorization PDP checks policy as to whether such external requests are allowed to be sent to the specified remote OIM instances, and informs the PEP of a yes/no. These steps are the same as 4-16 in the Authorize Client Action process described in Section 9.3.2.1.1, except that in Step 10-12, additional policy information is retrieved. This additional information is the set of attributes authorized to be sent to each target remote OIM, and mapping information if that OIM is not using the same attribute semantics.

20-25. Audit and situational awareness data is created based on each such authorization. These steps are the same as 17-22 in the Authorize Client Action process described in Section 9.3.2.1.1.

26. The PEP informs the Outgoing Remote Request Handler whether or not it can send each intended outgoing data request. For each permitted outgoing request, the PEP also provides the Outgoing Remote Request Handler the set of attributes authorized to be sent to that target remote OIM, and mapping information if that OIM is not using the same attribute semantics.
27. The Outgoing Remote Request Handler sends the request from the user to the PEP of the remote OIM. The request includes an assertion of the authentication of the user, the authentication assurance level of the user's authentication, the user's request, mapped versions of attribute values permitted to be sent to this remote OIM, and a certificate representing the originating user's OIM. All of this data is signed using the private key associated with this certificate.

9.4.2.1.3 Detailed Process Steps – Incoming Remote Request

This process flow details steps when an OIM PEP receives a request from a remote OIM instance.

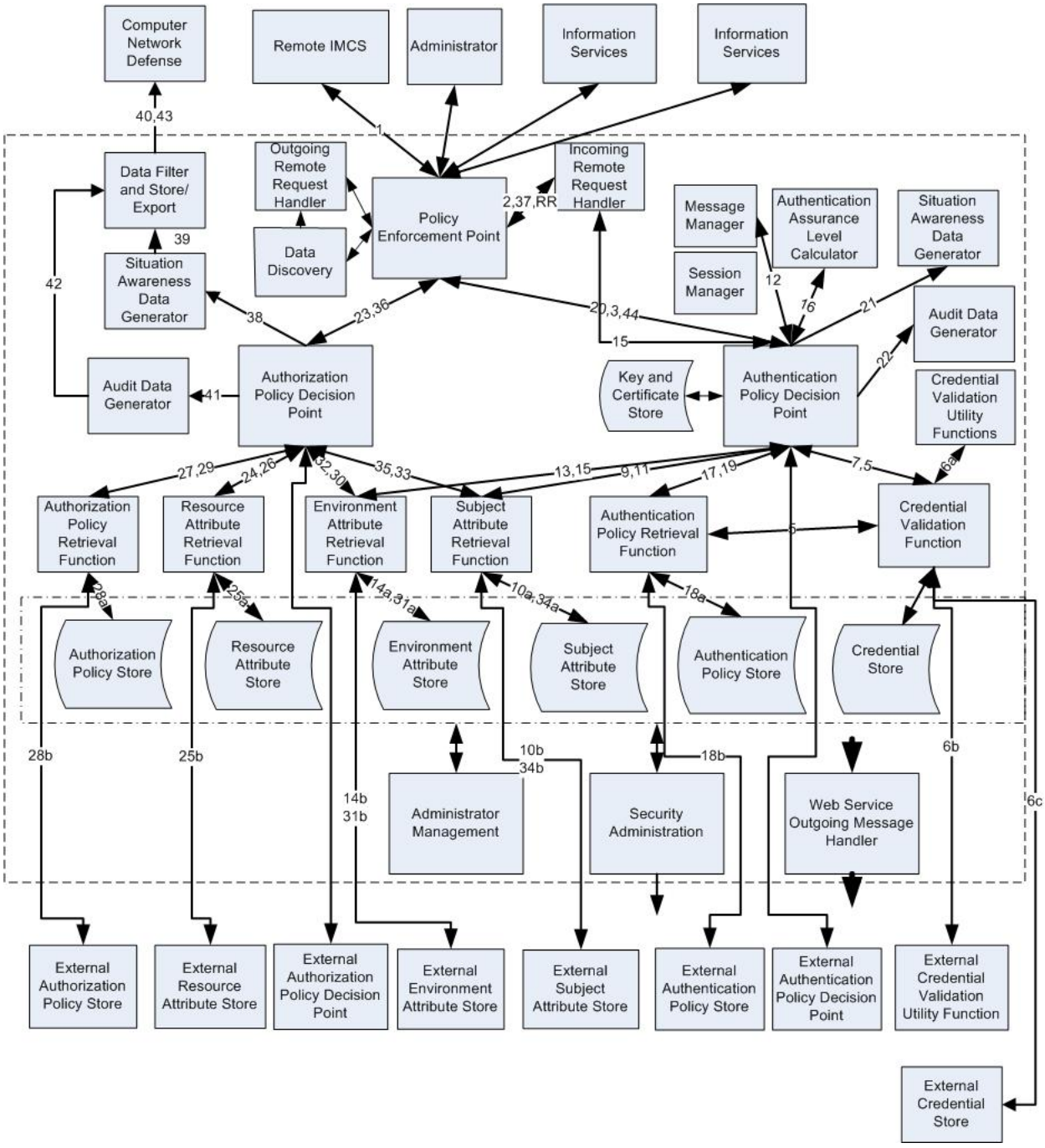

Figure 11 - Incoming Remote Request from Federated OIM

1. The PEP receives a data request from the Outgoing Remote Request Handler of a remote OIM instance.

2. The PEP provides the Incoming Remote Request Handler with information about the request such as IP address, that will be needed later to determine environmental attributes related to the requesting OIM.

3-22. The PEP first applies its authentication process to the requesting OIM, to authenticate the identity it presents. These steps follow the description of steps 3-23 in the CAC authentication flow in Section 9.2.2.1, omitting Step 4, and replacing Step 13 by a communication with the Message Manager instead of the Session Manager. Step 4 is not needed since no further authentication protocol interchanges occur between the two OIM's once the request is received. Also note that authenticating a remote system has the same differences from the user authentication process as noted under the process flow steps for Authentication of Automated Process as Client as described in Section 9.2.2.2.1. In particular, it is likely that the steps to consult the Subject Attribute Store, and the Authentication Assurance Level Calculator, are likely not to be implementable for an automated entity for some time, and thus would be skipped in the process flow.

23-36. If authentication succeeds, the PEP sends the remote request through the Authorization PDP, to determine if the system is allowed to process these kinds of requests from this requesting entity. The Authorization PDP returns a yes/no to the PEP. These steps are the same as 3-16 for Authorize Client Action in Section 9.3.2.1.1.

37. If this authorization succeeds, the PEP stores the data needed to respond to the remote OIM in the Incoming Remote Request Handler.

38-43. The system creates the required records about the authorization decision.

44. The PEP sends information about the client, their asserted authentication assurance level and the entity that sent the request, to its Authentication Policy Decision Point. The PEP asks the Authentication PDP to determine whether to accept the asserted authentication of the user.

Repeat 9,10,11,17,18,19,20,21,22. Based on attributes of the sending OIM as a subject, and the authentication policy, the PDP responds to the PEP that the authentication is accepted (which we assume in this case), and the system creates records about this decision.

Repeat 23-36. If the authentication was accepted then the PEP sends the user's request along to its Authorization PDP so that the user's authorization for the requested action can be checked. The policy is consulted to determine whether values of attributes asserted in the request can be accepted. Any attributes not received in the request, and required to evaluate the policy, are retrieved from the various attribute stores. The Authorization PDP returns a yes or no to the PEP.

Step RR. The PEP requests the Incoming Remote Request Manager to return a response to the client via the remote OIM that made the request.

Step not pictured. The Outgoing Remote Request Handler of the requesting OIM either fulfills the user's request or informs the user that this data in the federation that matches the request

could not be supplied. The request may be fulfilled by directly providing data to the client, or by providing the client a handle via which they may obtain the data directly from the OIM that manages it.

9.4.2.1.4 Related Issues

- **Authorization Policy Ownership** An open discussion point regarding this federation architecture is how to support the case in which a policy of the requesting IMCS may need to be applied to data that was obtained from a remote ICMS on behalf of a client. For example, if the requesting IMCS knows the user is in hostile territory, it may wish to deny delivery of certain kinds of data to that user. The remote federated IMCS may or may not know the circumstances of a particular user, so may not be able to apply such a policy.

  The related question in terms of requirements on federation, is determining who owns the authorization policy for a particular user related to a specific data object. There are several cases that might be supported:

  A) The data owner owns the policy. In this case, the policy is most naturally implemented in the remote IMCS that is managing the data.

  B) The supervisory organization that manages the user and their tasks, owns the policy. In that case, it is natural that the requesting IMCS, that originally logged in the user, has control over the policy. There are two subcases to consider here:

     - The requesting IMCS has some data related to the request
     - The requesting IMCS does not have any data related to the request.

     If the requesting IMCS has some data related to the request, it would make sense that this IMCS would have in place a policy that relates to this user's access to this type of data. In this case one might want to apply this policy also to data retrieved from a remote IMCS. If the requesting IMCS does not have any data related to the request, thus does not necessarily have related schemas implemented, it may be more problematic for the requesting IMCS to express and implement a policy regarding this data.

  C) Both the policies of the data owner and the supervisory organization for the user should be taken into account before returning data to the user. In this case policies maintained in both the requesting and remote IMCS need to be enforced. This case logically covers both cases A) and B) as well, so we discuss how it might be implemented.

  For case C), note that the policy held by the requesting IMCS can be enforced either:

  a) At the requesting IMCS, before sending the request to the remote IMCS

  b) At the requesting IMCS, after obtaining the data returned from the remote IMCS

  c) At the remote IMCS, by sending the policy to the remote IMCS along with the request.

The proposed architecture here contains a step in which the requesting IMCS checks its own authorization policy before sending an outgoing request to the remote IMCS in the federation. Although this policy was initially visualized to control remote requests at the level of what types of request may be sent to which federated partners, it could be extended to take into account the specific user making the request, and thus apply the requesting IMCS policy for the user before the remote outgoing request is ever sent. This is an implementation of case a) just above. A disadvantage of this approach is that the policy may change at the requesting IMCS between the time that the remote request is sent out, and the time the client receives the data from the remote IMCS. This is particularly apparent for deliveries on subscriptions.

The advantage of the approach in b) is that the policy applied after the data is returned is the most current policy, as opposed to the policy in force at the point when the remote request was sent out. A disadvantage is that data may be returned to the client via the requesting IMCS that the client cannot see, unnecessarily using up bandwidth and resources of both systems.

The approach in c) also has the disadvantage that the policy enforced on behalf of the requesting IMCS may be stale by the time data is returned from the remote system to the client. It also complicates the processing of the remote IMCS since it may encounter error conditions (unrecognized parameters for example) when attempting to apply a forwarded policy that was constructed by another member of the federation.

This topic requires more detailed study. Based on the analysis thus far described above, a reasonable approach would be for the requesting IMCS to apply the policy before sending the outgoing request, using the policy check already in the current proposed architecture, and then check a flag on the return of data to see if the policy has changed, and refilter the returned data if needed. This avoids the problem of using up unneeded bandwidth, and adds only a simple yes/no check on return of the data if in fact the policy has not changed, which will be the most common case.

- **Data Discovery Operation:** The operation of the Data Discovery module has not been described. It could for example, do a lightweight query of a pre-defined set of its "neighbors" to ask if they have anything relevant to this request, before sending the full request. Or, it might send the full request to everyone, and let them come back and say they don't have any relevant data. For queries, this finishes the story. For subscriptions, it is a bit more complicated – since even if the information services instance doesn't have any data matching the subscription now, it might later on, so may want to hang on to the subscription. An additional element of the Data Discovery module needs to determine the scope over which it will "look" for relevant data (which OIM instances will be considered in the search).
- **Security for Data Discovery:** The process by which the requesting OIM finds the OIM with the data the user wants, and the security that applies to that process, has not been addressed in this study. In particular, we have depicted "discovering" data in the federation and "asking" for this data as sequential steps, but as noted above this need not be the case. If these steps are combined for efficiency when both are needed, this essentially combines the Data Discovery and Outgoing Remote Request Handler functions in one module. However,

there would still be policies applied for each of these actions. In this case, the discovery and outgoing request policies would both obtained at once by this combined module from the PEP.

- **Audit for Federated Requests:** Although not detailed in the process flow, the sending OIM needs to audit what remote requests it sends out based on user's request, and responses received.

9.4.2.1.5 Decisions and Insights

- **No login to "remote OIM":** A user logs in to a single PEP. Their system handles hand off of requests for data that is not directly protected by the PEP to which they logged in. They do not log in to other PEP's to obtain access to data protected by those PEP's. Thus the complication of dealing with data in multiple OIM locations is masked as much as possible from OIM clients, and handled via server to server interactions.
- **Possible "partial" fill of query or subscription if data lives in multiple information services instances:** It is possible that that data relevant to a query or subscription is scattered over several OIM instances. In this case, it is also possible that some of the systems may respond to the remote request and some may not, as their policy may prohibit this.
- **No chaining of information requests:** In order to simplify authentication and authorization processing by the receiving system, we assume that it always receives requests directly from the system to which the user logged in, and not via a "chain" of requests that pass through multiple systems. This implies that the Data Discovery module associated with the user's PEP finds all ultimate sources for the data the user has requested. (However, the Data Discovery module might use a chaining process to *find* these locations.) This also implies that policies for acceptance of requests from remote systems need to cover all possible systems that might send such requests. This might appear to create an unmanageable policy for determining whether to accept remote requests. However, we assume that under the need to share concept, the most typical policy will be to allow all authenticated OIM instances to request data on behalf of their users. Thus management of the remote request policy reduces to managing exceptional cases where such requests will be rejected. The typical use of chaining of requests in a web services environment is for one service to invoke a different service (usually with a different owner) that it needs to perform or enhance its service, such as providing financial news to a brokerage client. In this case, there is only atomic service that is ultimately desired, which is the delivery of the information the user needs, by the owner of this information.
- **Original OIM responds to the client:** Our initial versions of this design had the target OIM responding to the client, due to the complications involved in having the original OIM respond to the client – in particular:
    - It may not be prudent for the initial OIM to handle all data returned to the client for performance reasons

- Tracking responses from remote OIM's for all remote requests sent out as a result of a client request, and then responding back to the client for each of these, determining what is missing, etc., is a significant job for the original OIM.

However, we judged that there were more complications involved if each remote target OIM responds directly to the client independent of the requesting OIM:

- There would be no way for the requesting OIM to abort remote requests if the client became unauthorized.
- The client would receive results from multiple end points, and not have the information required to understand whether all remote requests sent out as a result of their original request, had been responded to.
- If a user had a protocol such as an SSL tunnel set up with the initial OIM, that would be of no use when a remote OIM attempted to respond to them. Therefore the client might need to receive connection requests from several remote OIM's that it would need to authenticate, creating a complex situation for the client.

In summary, the decision of how responses to the client should be handled involves many issues beyond authentication and authorization. For authentication and authorization, the key issue identified is how to abort in process requests. Other non-security issues identified are performance impact on the requesting OIM, error identification and handling at the client, data in transit security.

10.0Module List

This section provides a reference list of the internal OIM modules in this architecture, and a brief description of their functions.

Section 0 provides an overview of these modules in the context of their interactions. Section 6.2 discusses the external entities with which OIM interacts.

10.1 Architecture Backbone

- **Policy Enforcement Point (PEP)** – is a single interface point for access to OIM system resources. A Policy Enforcement Point defines an "OIM instance", as we use that term here.
- **Authentication Policy Decision Point (PDP)** – determines whether an entity is authenticated.
- **Authorization Policy Decision Point (PDP)** – determines whether a specific access to a specific systems resource is permitted
- **Session Manager** – tracks attributes of in-process sessions
- **Message Manager** – tracks attributes of in-process message requests in web services mode, and requests from remote OIM instances

The architecture also allows for external PDP's, defined as follows.

- **External Authentication Policy Decision Point (PDP)** – is an externally implemented decision logic module that determines whether an entity is authenticated based on a subset of all attributes relevant to the authentication attempt. The internal PDP may consult this external resource for a "sub-decision" relevant to its ultimate authentication decision.
- **External Authorization Policy Decision Point (PDP)** – is an externally implemented decision logic module that determines whether a specific access to a specific systems resource is permitted based on a subset of all attributes relevant to an access attempt. The internal PDP may consult this external resource for a "sub-decision" relevant to its ultimate access decision.

Examples of these would be PDP's that provide decisions based on a community of interest policy. Thus OIM does not need to implement the COI policy internally, but can request a decision externally, and integrate that decision with its internal policies.

10.2 Authentication Decisions

Attributes stores may be available from GIG services that are external to OIM. The retrieval functions listed incorporate the logic for retrieving both required OIM internal and external data.

- **Authentication Policy Retrieval Function** – retrieves internal and external policies that determine what authentication results are returned by the Authentication PDP
- **Authentication Policy Store** - stores internal and external policies that determine what authentication results are returned by the Authentication PDP. These are policies that determine whether authentication "passes," for reasons above and beyond the specific authentication protocol completing successfully. Some examples of such policies are:
    - Allowable authorities for verifying credentials or obtaining subject attribute data
    - Policy that says that authentication is denied for all requests coming from a particular network known to be under hostile control.
    - Policy that says that only certificate-based authentication of a certain strength is accepted by this system.
    - Policy that says only authentications above a certain authentication assurance level are accepted, unless they are coming from the local network.
    - Policy that determines whether client attributes sent as part of a request from a remote federated OIM will be accepted.
- **Environment Attribute Retrieval Function** – retrieves attributes of the OIM and client environments, and external attributes such as DEFCON level, that may be taken into account by the Authentication PDP and Authorization PDP.
- **Environment Attribute Store** – stores attributes of the OIM and client environments such as security posture of client systems and intervening networks, and processing load of the OIM server itself. It also stores external attributes of the battlespace environment such as DEFCON level that may be taken into account by the Authentication PDP and Authorization PDP.
- **Subject Attribute Retrieval Function** – retrieves internal and external attributes of entities such as human users and automated processes that may attempt access to the system
- **Subject Attribute Store** - stores internal and external attributes of entities such as human users and automated processes that may attempt to access the system. Examples are identity, identity proofing strength of mechanism, GIG user ID(s), active/suspended status as GIG user, clearance level, nationality, roles, COI memberships.
- **Credential Store** – stores credentials for those authentication decisions that require credentials for reference. For example, a certificate based protocol may send either the certificate itself, or a reference to it, in the client request. If a reference is used for efficiency at the client, OIM will need to look up the certificate, most likely in an external certificate store such as the NCES Certificate Retrieval Service. An example of an internal OIM credential store is the local password database.
- **Authentication Assurance Level Calculator** – calculates authentication assurance level

- **Credential Validation Function** – coordinates processing steps to determine whether a presented credential is valid
- **Credential Validation Utility Function –** performs processing steps required to validate a specific credential type, such as certificate path validation, digital signature checking or password matching

10.3 Authorization Decisions

Note that environment attributes, subject attributes and authentication assurance level are used for both authentication and authorization decisions. The modules related to these are described above in Section . Here we list additional modules involved in authorization decisions.

- **Authorization Policy Retrieval Function** – retrieves internal and external policies that determine what authorization results are returned by the Authorization PDP
- **Authorization Policy Store** – stores internal and external policies that determine what authorization results are returned by the Authorization PDP. Examples are:
    - Allow a user in the Planner role access to modify information objects with metadata that indicates they are "plans"
    - Allow a specific automated process to publish data objects whose metadata indicates they are situation updates
    - Allow processing of a federated request for any information if it comes from the OIM at a specific friendly base
- **Resource Attribute Retrieval Function** – retrieves attributes of resources (both information objects and administrative data) that may be taken into account by the Authorization PDP
- **Resource Attribute Store** – stores attributes of resources (both information objects and administrative data) that may be taken into account by the Authorization PDP**.** For OIM, this is typically metadata. External stores may contain attributes that apply to general categories of data, where the categorization is based on metadata characteristics, but these attributes are not directly realized as metadata in OIM. Examples are releasability and quality of protection based on category of data.

10.4 Federation

- **Data Discovery** – determines whether a request requires services from a remote federated OIM, and locates this OIM.
- **Outgoing Remote Request Handler** – packages and sends requests to remote federated OIM instances to satisfy requests for clients logged into this OIM instance
- **Incoming Remote Request Handler** – coordinates processing of requests arriving from remote OIM instances on behalf of clients logged in to those systems.

10.5 NCES Compatible Web Services

- **Web Services Outgoing Message Handler -** packages outgoing messages to NCES-compatible web services in the NCES-defined format.

10.6 Keys and Certificates

- **Key and Certificate Store -** contains the private key and certificate for this OIM instance. The Key and Certificate Store also may contain a pre-distributed NCES Certificate Validation Service certificate.

10.7 Administration

- **Security Administration** - provides the management interface to the data in the various stores noted above.
- **Administrator Management** - provides a management interface to set up the privileges for the various administrators of the system.

10.8 Data Generation for Computer Network Defense

- **Audit Data Generator** – generates audit records for authentications performed and accesses to resources attempted, rejected, and granted.
- **Situation Awareness Data Generator** – provides data for a real time or near real time analysis process that monitors the security health of OIM, and of the GIG or portions of the GIG.
- **Data Filter and Store/Export Function** - determines how and when to filter, format and store and/or forward situation awareness data and audit data to internal stores and external recipients.

10.9 Multiple Instances of OIM Modules

The following table highlights those modules for which more than one module instance may appear in an OIM instance as defined in this architecture. If there can be multiple instances of a module, this implies that others modules with interfaces pointing to this module must include intelligence to determine which instances to communicate with.

Table 11 - Multiple Module Instances in an OIM Instance

Module	Multiple Instances Possible?	Comments
Policy Enforcement Point	No	An OIM instance has by definition a single PEP. In other words, an OIM instance includes all information protected by a single PEP.

Authentication Policy Decision Point	No (int) Yes (ext)	A single internal PDP for authentication supports the PEP. There may be multiple external PDP's assisting the internal PDP in authentication decisions.
Authorization Policy Decision Point	No (int) Yes (ext)	A single internal PDP for authorization supports the PEP. There may be multiple external PDP's assisting the internal PDP in authorization decisions.
Session Manager	No	
Message Manager	No	
Authentication Policy Retrieval Function	No	
Authentication Policy Store	Yes	There may be various kinds of policy in various stores, both internal and external. It is the job of the retrieval function to find the required policies.
Environment Attribute Retrieval Function	No	
Environment Attribute Store	Yes	There may be a number of internal and external stores with different kinds of environment attributes. For example, one might imagine an internal store for site INFOCON level, an internal store for status of connected networks, an external store for security posture of connecting devices and an external store for INFOCON levels of other sites. It is the job of the retrieval function to find the required attributes.
Subject Attribute Retrieval Function	No	
Subject Attribute Store	Yes	There may be a number of internal and external stores for subject attributes. There may be separate subject attribute stores for users in different identify management domains, and for human users vs. devices or processes.
Credential Store	Yes	There may be any number of internal and external credential stores, each of which contains different types of credentials for different subsets of subjects.
Authentication Assurance Level Calculator	No	Although different algorithms may be used in different cases, the decision of which one to use remains part of the single Authentication Assurance Level Calculator module.

Credential Validation Function	Yes	There may be several internal and external instances of these functions, one for each type of credential supported by the OIM instance.
Credential Validation Utility Function	Yes	There may be any number of internal and external utility functions, as required to support validation of each type of credential supported by the OIM instance.
Authorization Policy Retrieval Function	No	
Authorization Policy Store	Yes	There may be various kinds of policy in various stores, both internal and external. It is the job of the retrieval function to find the required policies.
Resource Attribute Retrieval Function	No	
Resource Attribute Store	Yes	Internal resource attributes are metadata in OIM. As shown in Figure 1, a single PEP may protect more than one data repository, so there may be more than one metadata store per OIM instance. There may also be more than one external store for resource attributes, for example one for quality of protection attributes and one for releasability attributes.
Data Discovery	No	
Outgoing Remote Request Handler	No	
Incoming Remote Request Handler	No	
Web Services Outgoing Message Handler	No	
Key and Certificate Store	No	
Security Administration	No	
Administrator Management	No	
Audit Data Generator	No	
Situation Awareness Data Generator	No	
Data Filter and Store/Export Function	No	

11.0 User Attribute Store

The following example illustrates the GIG identity concepts by showing what a record for one individual might look like.

Note the following features of this example.

- The record includes descriptive data about the individual including address phone, clearance.
- The individual may have more than one *GIG identity identifier*. This is a list of attributes issued by a particular *management authority* that uniquely identifies this individual. It is not used for login, and remains constant once issued.
- There will be a standard metric for the strength of the process that an ID management authority uses to verify the identity of this individual. This is called the *ID Proofing Score*.
- The GIG user ID is used for login to the GIG. A GIG user ID is bound to one identity identifier. The GIG user ID is prepended with the ID of the issuing management authority to insure uniqueness across the set of GIG user IDs. An individual may have more than one GIG user ID, though this will not be the usual case. The example shows a GIG user ID associated with each identity identifier for the individual. A GIG user ID has a *status* such as active/suspended.
- *Roles* recognized across the GIG as allowing the user specific types of access are part of the record.
- In addition, the user may belong to *Communities of Interest* (COI's) that will afford them access to particular kinds of information based on policy.
- Finally, the record contains a pointer to the credentials this user is using. This is for the purpose of supporting maintenance of these credentials, for example, revoking the user's certificate if the associated GIG user ID is not longer active. Credentials are bound to a particular GIG user ID. Although not shown in the example, more than one credential may be associated with a single GIG user ID.

Example Record:

```
Address: 8926 Big Tree Way, NY, NY
Phone: 000-000-0000
Clearance = TS
```

Table 12 - GIG Identification Data

GIG Identity Identifier (attribute list)	ID ID Proofing Score	GIG User ID		GIG User ID Status	Roles	Other Attribute or Privilege	COIs	Credentials
		ID Mgmt Authority ID	Identifier					
NY; Lic 899333	5	DEERS	JDoe	Active	ACCTNG			Password Hash
Empl 855; BDAY 5/5/50	10	IC	JDoe	Active	FINANCE-SUPV	WEEKEND ACCESS	FRAUD COI	PKI Cert 1
SSN 100000 00	8	Commercial	JohnDoe	Deactivated	ACCTNG CONTR			PKI Cert 2

12.0Pluggable Authentication Architecture

Figure 12 illustrates how one might construct the OIM modules shown to support simple plug-in of new authentication methods. Note that the Decision Logic submodule supports the policy-based consideration of characteristics of this event beyond the success of the protocol, such as the environmental situation of the client or a general policy on what strengths of authentications to accept.

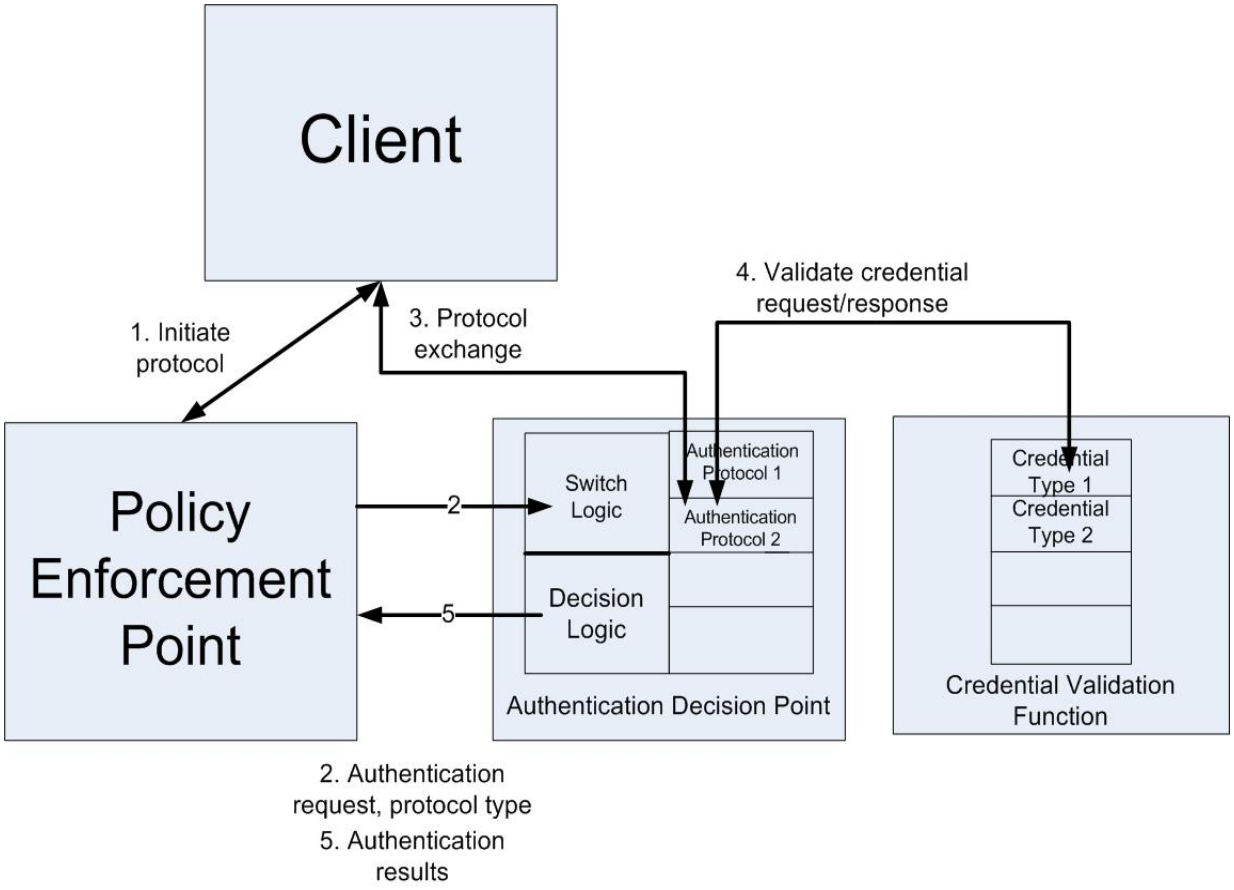

Figure 12 - Pluggable Authentication Architecture

13.0Metrics and Risk Analysis

Mission Oriented Risk and Design Analysis provides a way to model threats against the OIM SA (or any other system). By understanding all the different ways in which a system can be attacked, countermeasures can be designed or implemented to thwart them. This analysis also provides a way to rank all the different ways in which a system can be attacked, so that the level of effort and resources expended on the countermeasures are commensurate with the real level of threats.

The process consists of six steps which are described in the subsections below.

- Characterization of assets
- Characterization of adversaries
- Assignment of weights for cost, difficulty, detection
- Identification of attack goals
- Development of attack trees (strategies)
- Application of the weights to the attack trees to determine scores

The steps do not have to be done in this order, and some steps are reusable. Although this work has been on an authentication and authorization architecture for the OIM, the overall attack goals, and therefore the information assurance concerns for a system, encompass more than just authentication and authorization.

13.1 Asset Characterization

Asset characterization involves enumerating not only what the components of the system are, but also what value they have to the mission. At the architecture level, assets are the data stores, the flows and the processing components that must be protected. The identification of mission-critical assets forms a basis for the identification of overall attack goals.

13.2 Adversary Characterization

It is necessary to understand who the attackers are, and to consider their abilities, motivations, and goals to be able to identify and quantify the real threats. Adversaries are the “what” and the “who” that the OIM SA must be defended against. Different types of adversaries will have different goals, and will also respond differently to countermeasures.

Types of adversaries include:

- A foreign government or military
- An economic or business competitor
- A terrorist organization

- Organized crime
- Hackers
- Insiders (malicious)
- Unintentional misuse

The last category of adversary, unintentional misuse, is included to account for risks to the OIM SA from user errors, carelessness, and failure to follow prescribed procedures.

The list of adversary types could be expanded or changed for a different risk analysis:

- Press
- Law enforcement
- Activists
- Lone criminals
- Script kiddies

13.3 Weighting

The weighting step assigns numeric values to the adversary characterizations in three dimensions: difficulty, cost, and sensitivity to detection. An example is shown in Figure 13.

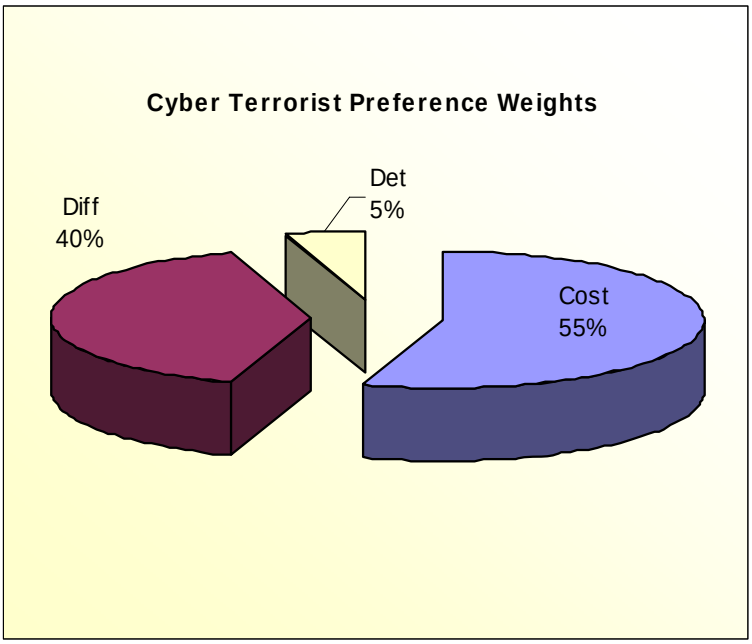

Figure 13 - Adversary Preferences

In addition to the overall proportions, the adversary's preferences are plotted on utility curves, such as the one shown in Figure 14. This utility curve plots the change in probability that the adversary will undertake an action as the cost increases.

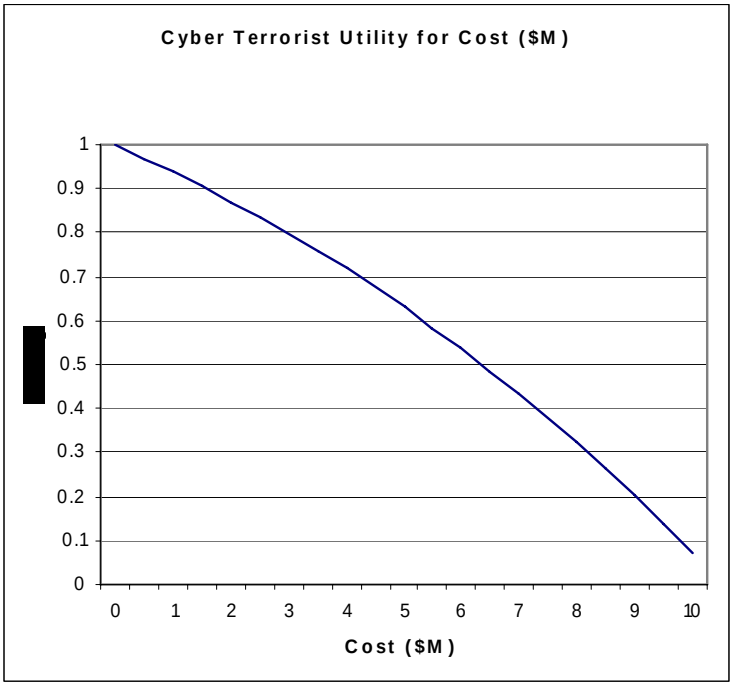

Figure 14 - Example of a Utility Curve

The preferences and utility curves will change depending on the situation. For example, if a foreign government is at war, it may be less concerned about being detected than it would be during peacetime. The shape of the utility curves will also depend on the environment being modeled, for example, access to communications infrastructure may be more difficult in a permanent headquarters than in a deployed unit.

13.4 Identify Attack Goals

At the basic level, any attack has a single goal – interruption of mission. Generally, attacks are categorized into three subsets:

- Breach of Confidentiality
- Breach of Integrity
- Breach of Availability

For example, fraudulent access (subscribe or query) via interception or other means would be a breach of confidentiality. Fraudulent publishing (spoofing) would be a breach of integrity. A denial of service attack would result in a breach of availability.

13.5 Develop Attack Trees

Attack trees provide a formal, methodical way of describing the security of systems, based on different attacks. The attacks against a system form a tree structure, with the goal as the root node and different ways of achieving that goal branching out into leaf nodes.

An attack tree defines and sequences the individual steps needed to reach a goal. Some of the steps may be implementation-dependent, and so can not be defined at the architectural level. But they can be accounted for, and the effects at the architectural level can be modeled.

To illustrate, Figure 15 shows an example of a simple attack tree for gaining access to email.[19]

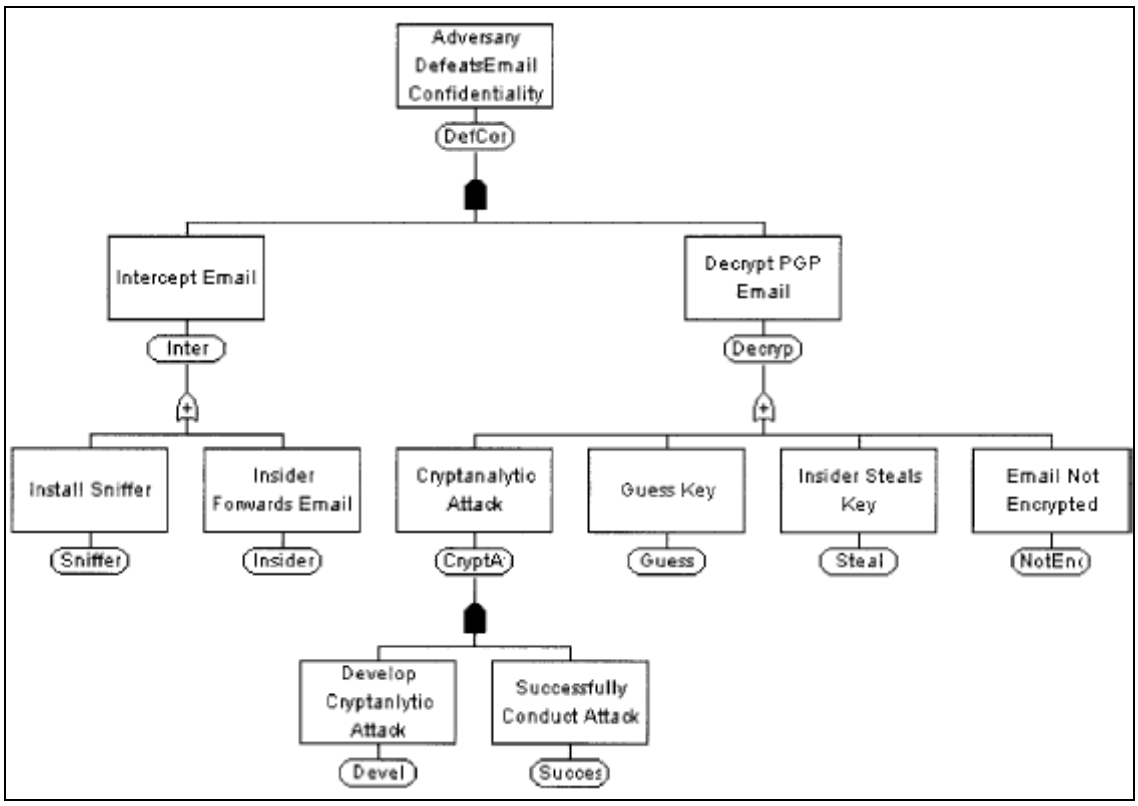

Figure 15 - Example of an Email Attack Tree

[19] Buckshaw, D. L., G. S. Parnell, W. L. Unkenholz, D. L. Parks, J. M. Wallner, and O. S. Saydjari (2005). "Mission Oriented Risk and Design Analysis of Critical Information Systems", *Military Operations Research*, V. 10, N. 2: 19-38.

The root node (goal) is fraudulent access to email. There are two sub-goals required to achieve the root goal: 1) to intercept the email and 2) to decrypt it. On the left branch of the tree are shown two possible ways that an adversary could achieve the sub-goal of intercepting email: 1) install a sniffer or 2) receive it from an insider. Similarly, on the right are the various steps needed to decrypt email, including the possibility that the email is unencrypted.

13.6 Scoring

The formula for the Mission Oriented Risk and Design Analysis (MORDA) risk calculation is shown in Figure 16. The R_a is calculated for each combination of adversary and attack.

$$R_a = I_a[w_{cost} \cdot U(cost_a) + w_{diff} \cdot U(diff_a) + w_{det} \cdot U(det_a)]$$

Where:
a = system attack index
R_a = overall risk score of attack a

Parameters:
$cost_a$ = cost of attack a
$diff_a$ = degree of difficulty of attack a
det_a = level of detectability of attack a
I_a = Expected impact to the user's mission if attack a is successful

Weights:
w_{cost} = weight of cost parameter
w_{diff} = weight of difficulty parameter
w_{det} = weight of detectability parameter

Utilities:
$U(cost_a)$ = the adversary's utility for attack a derived from the cost parameter
$U(diff_a)$ = the adversary's utility for attack a derived from the difficulty parameter
$U(det_a)$ = the adversary's utility for attack a derived from the detectability parameter

Figure 16 - MORDA Scoring

Each "attack" (R_a) is one possible path through the attack tree from leaf to goal. The parameters, weights and utilities are applied to each step in an attack tree path. This gives a risk score for each attack. Notice that the mission impact (I_a) is a factor in each attack score. This ensures that hardening or mitigation efforts are applied to the areas that matter most.

There is necessarily much analysis that must be done in order to get the numbers needed to plug into the formula. The better and more precise this part is, the more useful the results will be. There are also parts that may be completely arbitrary. Since the goal is to come up with relative

scores (likelihood of this attack vs. that attack), or more usefully, the inclusion of this architecture component vs. that architecture component, as long as the arbitrary decisions are made consistently across the contenders, whether or not the final "score" is "absolute" does not matter.

13.7 Analysis of Results

The results of applying a MORDA analysis can determine which attacks against the architecture have the highest risk. Knowing that, the architecture can be modified, or countermeasures added.

The analysis can also be used to compare the security risk for this architecture to that for other proposed architectures, or to compare alternatives within this architecture. The analysis, however, can not provide a single score that indicates the risk of compromise of the architecture, or of the OIM. That is beyond state of the art.

13.8 OIM SA

The risk scores by will be affected by the different configurations of environment and component type that the OIM encompasses. Environments include strategic, mobile, and portable. The component types include client, server, and centralized services.

Without undertaking a complete MORDA analysis of the proposed security architecture for the OIM, which was not in the scope of work, the some conclusions can be made based on a MORDA-like analysis of the OIM SA. For example, a possible attack against this architecture would be to attempt to use the federated communication mechanism to gain unauthorized information. Another example would be an attempt to make unauthorized changes to a policy. Further, the analysis helps to pinpoint needed mitigations or countermeasures. For example, a path through an attack tree relies on an adversary gaining access by presenting an expired credential. By adding a check of the certificate revocation list to the credential check function, the likelihood of someone gaining access with an expired credential is reduced.

14.0 Recommendations for Future Research

Recommendations for future research to extend the security architecture are described next.

14.1 Federation

14.1.1 Lightweight vs. Heavyweight Federations

This architecture defined a lightweight "pairwise" version of federation, where no new modules external to the federated OIM instances need be set up in order to initiate and operate a federation. Future study would examine scalability of this model for large federations and consideration of a more heavyweight model that centralizes federation configuration and policy. Hybrid models may also be useful, that centralize some aspects of federation configuration and policy but not all.

Since the success of a federation may depend upon how efficiently it can be formed and how quickly it can evolve, future work would study the steps to initiate and modify a federation based on the model presented in this architecture. This perspective may lead to improvements or refinements of the architecture. In addition, this topic should be considered in conjunction with the previous topic of lightweight vs. heavyweight federation.

14.1.2 Security for Detailed Federation Operations

Study of the following security topics relevant to federation would be best accomplished in conjunction with design of the underlying functional requirements and capabilities:

- Security for discovery
- Return of data to the client from multiple federation members
- Canceling subscriptions or queries in-process in the federation based on logout or permission change of the ultimate requester.

14.1.3 Process Flows for Additional Federation Scenarios

Additional process flows have been identified for which development would validate and/or refine the architecture. In particular:

- Detail process flows for GIG I&A Alternative 3 as mentioned in Section 7.1.1.1.3 to verify architecture support for these operations. The two specific cases mentioned are:
    - a user interacting with a web portal, that in turn interacts with OIM using a non-web-service interface

- a user interacting with an NCES thin client (such as a web portal), that in turn interacts with OIM using a NCES compatible web service interface.

- Develop federation process flow case in which the OIM that receives a federation request does not accept authentication from the requesting OIM.

14.2 Policy Ownership, Management, Reconciliation

In Sections 6.4.3 and 6.4.4, we put forth the concepts of "choose authority" algorithms and "reconcile authority" algorithms as a way of thinking about how to automate integration of interacting policies. Insight into useful algorithms could be gained from developing a number of case studies that involve internal/external policy interactions and potential conflicts among federation members.

14.3 Data Replication for Performance

An overall analysis of those areas likely to require caching or other data replication strategies would be of interest. We have noted some areas in the architecture where caching is likely to be required for adequate performance. For example, we noted a need to cache authentication and authorization information when operating in web services mode. As a second example, GIG studies discuss local stores replicating data in external stores for performance reasons, and a general framework to support this could be added to the architecture.

14.4 Security Administration

Future work would develop security administration process flows, and add detail to the architecture to support this functionality. This includes the impact of administrative changes on in-process transactions. Authorization specifically is unique for administrative actions since authorization in this case does not consider metadata, thus developing this area may introduce new concepts in the architecture.

14.5 MLS

Future work would study the impact on the federation concept presented in this architecture when members of the federation are at different security levels and connected via a guards.

14.6 Architecture Refinement

Section 2.3 described the process to develop this architecture, which included generation of rough requirements, identification of process flows and mapping to these requirements. Future work could formalize this effort further, in particular:

- Formalize the set of requirements, and complete the set of process flows

- Complete map of requirements to process flows

- Map requirements to architecture
- Develop process flows not addressed in this phase
- Add additional data item level detail to the architecture
- Develop functional definitions of the OIM modules described here, based upon their operations as described in the functional flow diagrams.

15.0 Conclusions

This study produced an architecture that met its identified driving requirements (Sections **Error! Reference source not found.** and 7.3). It includes a larger number of modules and interfaces than may have been anticipated. This was a result of the goal to build in flexibility for a variety of applications of OIM in a number of environments. Thus any particular instance of OIM will include a subset of these modules and interfaces.

The study details the options available to an individual OIM application and how these options are realized in the architecture (Section 6.4). Options available include choice of authentication protocol, use of internal and/or external sources for policies, attributes, and policy decision functions, flexible set of policy parameters, flexible definition of authentication assurance level, configurable targets and format for delivery of audit and situation awareness data, web services or non-web services client interface, and federated or stand-alone operations.

The architecture supports authentication and authorization for both a non-web service mode and an NCES compliant web services interface to clients with a nearly identical process flow, by treating an NCES compliant service request as an authentication credential, and its validation as an authentication protocol (Section 9.2.2.4).

This work yielded the following insights:

- **Supporting automated clients:** The same overall authentication and authorization process flow works for human and automated clients, but the underlying identity, credential and policy infrastructures to support this flow will exhibit many differences. Further, the infrastructure for automated clients is immature to non-existent at this time (Section 9.2.2.2).

- **Impact of NCES support:** Analysis of the impact of NCES support showed that it takes nearly the same amount of work on security features to take advantage of NCES Security Services as to provide a web service interface to clients (Section 6.3.6)

- **Applicability of commercial federation architectures:** Many of the tough problems faced in the commercial federation environment are not present in the DoD/GIG environment, thus a large body of commercial work in this area is not applicable to the problem at hand. However, commercial architectures for single sign on and attribute mapping were applied to this architecture (Section 7.1.4.3).

- **Tough problems in authorization:** The architecture for authorization with modules for policy enforcement, policy decisions and underlying attribute data is quite standard. This work borrowed the insight from the KAoS work that the policy evaluation process must interleave data retrieval and policy evaluation when the set of attributes is dynamic, rather than perform these steps sequentially (Section 7.1.3). Resolving lower level details that treat ownership, management and reconciliation among various sources of policy and decision

makers is much harder. The start of a conceptual framework for this and its relationship to this architecture is presented in Section 6.4.3.

The OIM security architecture was able to use and/or add supporting architectural detail to GIG and NCES architecture concepts to meet its needs, with two exceptions, which may well be treated in future work on these architectures:

- At this stage the GIG has not considered the use of interacting external and internal policy decision points, which integrates a hierarchy of policy decisions vs. a flat decision architecture based directly on attributes (Section 7.1.1.2)

- NCES service-oriented authorization architecture does not cover access control at the data layer required by OIM to control access to individual information objects or categories of information objects defined by metadata expressions (Section 7.1.2).

Major functional areas not covered by the study were fuselets, security administration, security for discovery in a federation, and the impact of multi-level security on the architecture. Section 8.0 outlines issues for MLS. Future architecture work is suggested in Section **Error! Reference source not found.**.